Secure Console Port Server SSH
Installation & Service Manual

Black Box

Secure Console Port Server SSH Installation & Service Manual

Version 1.3.3 release 1 – July 2002
Copyright (C) Black Box, 2002

FCC Warning Statement:
The Secure Console Server SSH has been tested and found to comply with the limits for Class A digital devices, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the Installation & Service Manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user is required to correct the problem at his or her own expense.

Notice about FCC compliance for the Secure Console Server SSH 16-Port and the Secure Console Server SSH 32-Port:
In order to comply with FCC standards the Secure Console Server SSH 16-Port and the Secure Console Server 32-Port require the use of a shielded CAT 5 cable for the Ethernet interface. Notice that this cable is not supplied with either of the products and must be provided by the customer.

Canadian DOC Notice:
The **Secure Console Server SSH** does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le **Secure Console Server** n'émete pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la classe A prescrites dans le règlement sur le brouillage radioélectrique edicté par le Ministère des Communications du Canada.

Table of Contents

CHAPTER 1 HOW TO USE THIS MANUAL

This manual assumes that the reader understands networking basics and is familiar with the terms and concepts used in Local and Wide Area Networking. The Secure Console Port Server SSH is a Linux-based terminal server, which gives it great flexibility. It runs an embedded version of the Linux operating system and Unix and Linux users will find the configuration process very familiar. On the other hand, users not familiar with Unix will have a steeper learning curve, but it is not necessary to be a Unix expert.

Configuration of the equipment is done by editing a few plain-text files (commented sample files for the principal profiles are provided in appendix C), and then updating the versions of the files in the Secure Console Port Server. The files can be edited in the Secure Console Port Server using the vi editor provided, or in another computer with the environment and text editor of your choice. Unix user or not, we strongly recommend that you follow the steps in this Installation & Service Manual before jumping in.

This manual should be read in the order written, with exceptions given in the text.

CHAPTER 2 SAFETY INSTRUCTIONS

Use the following safety guidelines to protect yourself and your Secure Console Port Server SSH.

USING YOUR Secure Console Port Server SSH

CAUTION: Do not operate your Secure Console Port Server SSH with the cover removed.

- In order to avoid shorting out your Secure Console Port Server SSH when disconnecting the network cable, first unplug the cable from the equipment and then from the network jack. When reconnecting a network cable to the equipment, first plug the cable into the network jack, and then into the equipment.
- To help prevent electric shock, plug the Secure Console Port Server SSH into a properly grounded power source. The cable is equipped with a 3-prong plug to help ensure proper grounding. Do not use adapter plugs or remove the grounding prong from the cable. If you have to use an extension cable, use a 3-wire cable with properly grounded plugs.
- To help protect the Secure Console Port Server SSH from electrical power fluctuations, use a surge suppressor, line conditioner, or uninterruptible power supply.
- Be sure that nothing rests on the cables of the Secure Console Port Server SSH and that they are not located where they can be stepped on or tripped over.
- Do not spill food or liquids on the Secure Console Port Server SSH. If it gets wet, contact Black Box.
- Do not push any objects through the openings of the Secure Console Port Server SSH. Doing so can cause fire or electric shock by shorting out interior components.
- Keep your Secure Console Port Server SSH away from heat sources and do not block cooling vents.

WORKING INSIDE THE SECURE CONSOLE PORT SERVER SSH

NOTICE: Do not attempt to service the Secure Console Port Server SSH yourself, except following instructions from Black Box Technical Support personnel. If this is the case, first take the following precautions:

· Turn the Secure Console Port Server SSH off.

· Ground yourself by touching an unpainted metal surface on the back of the equipment before touching anything inside it.

REPLACING THE BATTERY

A coin-cell battery maintains date and time information. The Secure Console Port Server SSH 1-Port does not have the battery, so the date and time must be kept up to date by ntpclient. If you have to repeatedly reset time and date information after turning on your Secure Console Port Server SSH, replace the battery.

CAUTION: A new battery can explode if it is incorrectly installed. Replace the 3 Volt CR2032 battery only with the same or equivalent type recommended by the battery manufacturer. Discard used batteries according to the battery manufacturer’s instructions.

CHAPTER 3 WHAT IS IN THE BOX

The Secure Console Port Server SSH is a line of console access and terminal servers. There are several models with differing numbers of serial ports. The following figures show the main units and accessories included in each package and how cables should be connected.

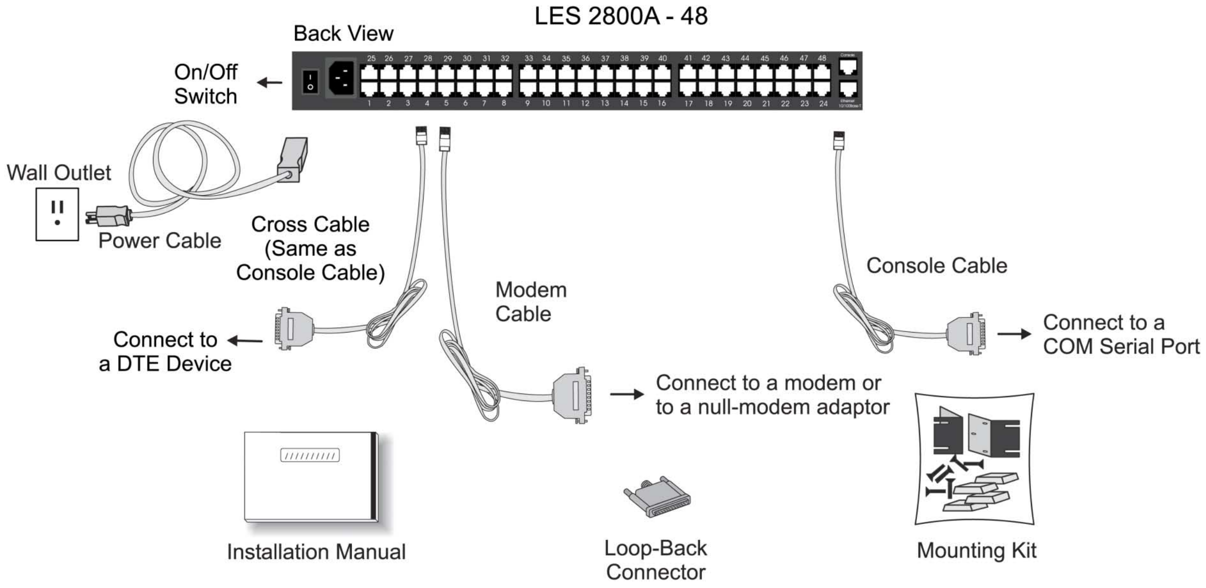


FIGURE 2.1 SECURE CONSOLE PORT SERVER SSH 48-PORT AND CABLES

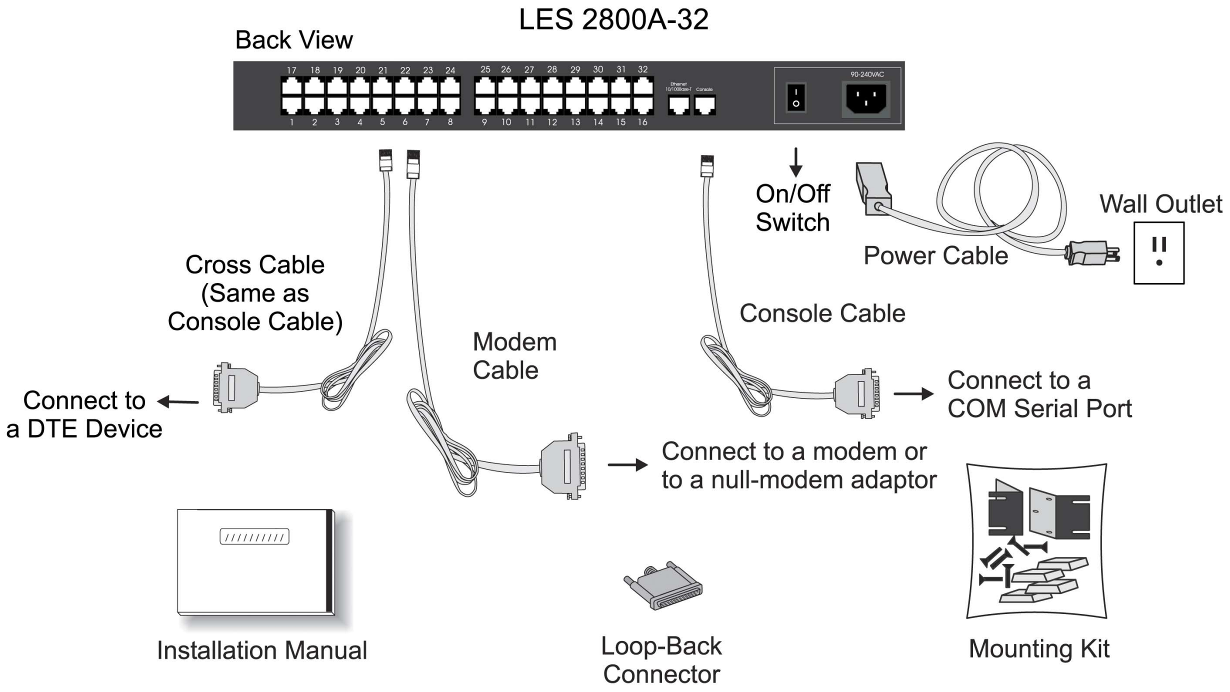


FIGURE 2.2 SECURE CONSOLE PORT SERVER SSH 32-PORT AND CABLES

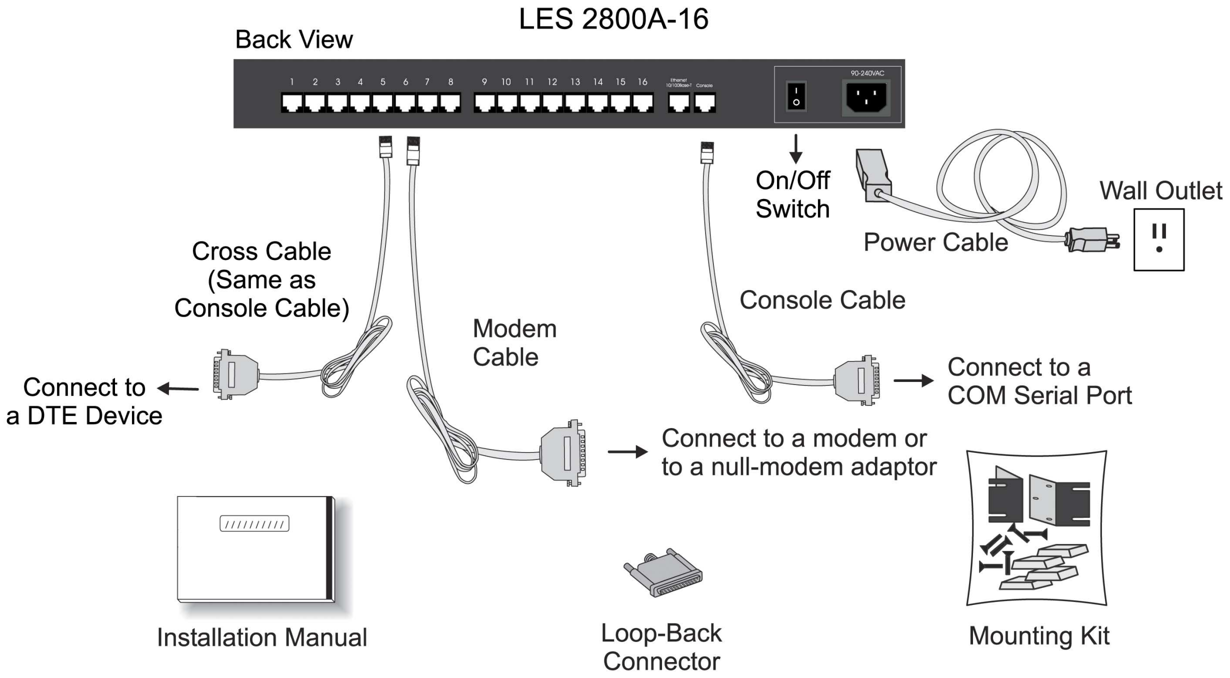


FIGURE 2.3 SECURE CONSOLE PORT SERVER SSH 16-PORT AND CABLES

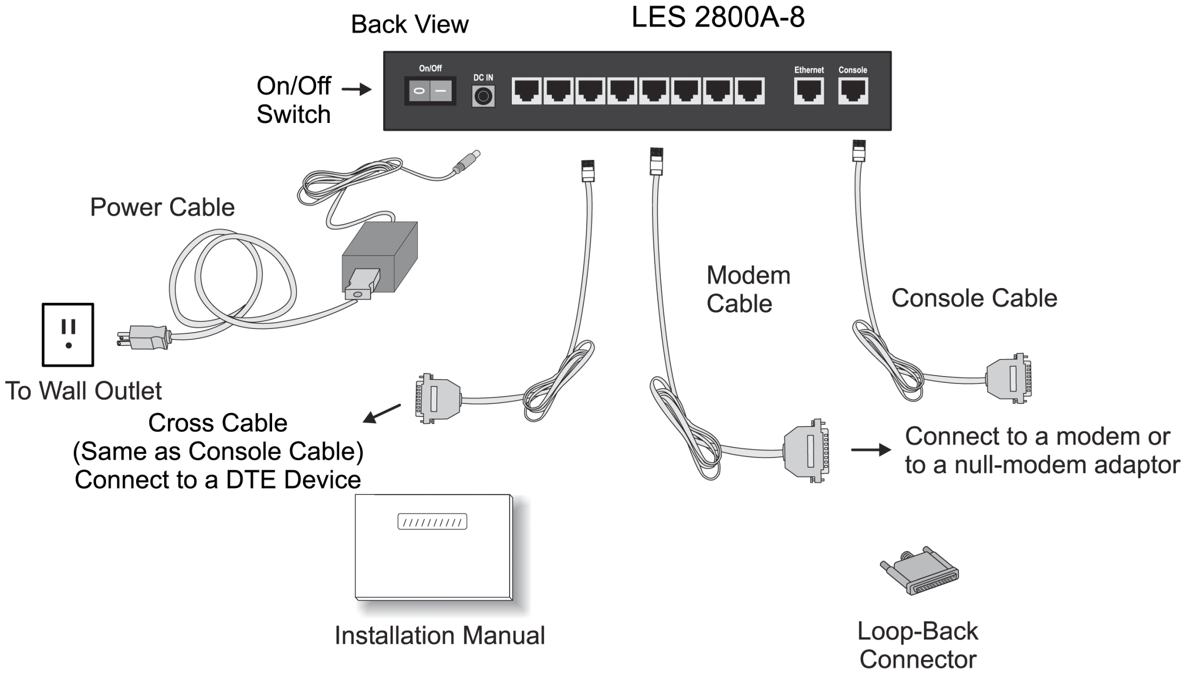


FIGURE 2.4 SECURE CONSOLE PORT SERVER SSH 8-PORT AND CABLES

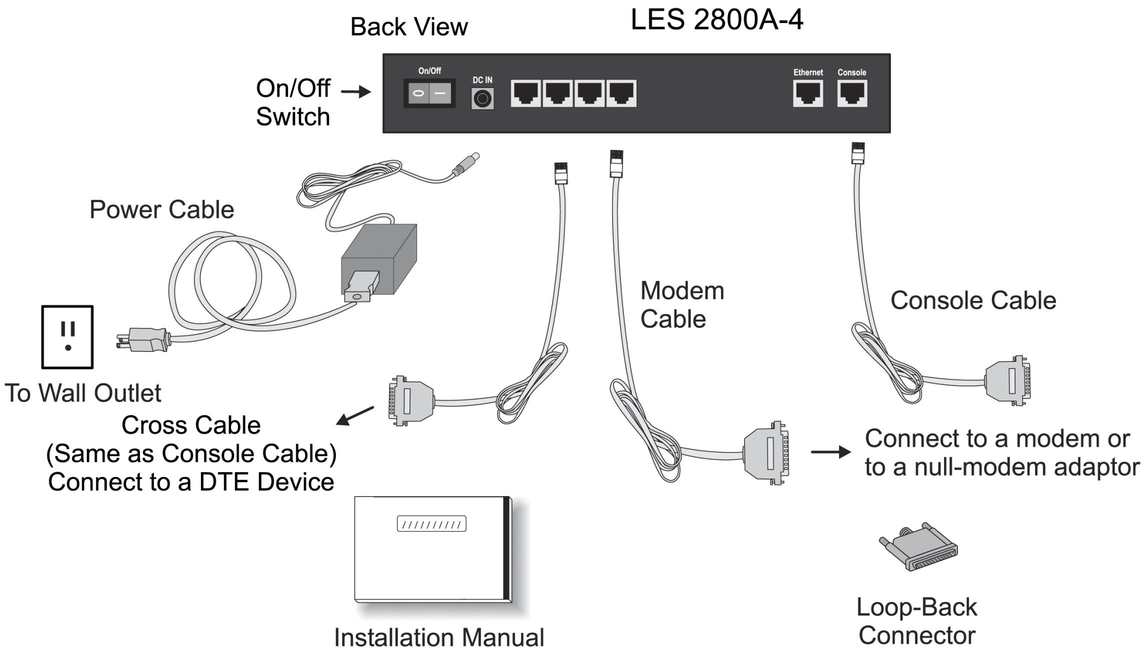


FIGURE 2.5 SECURE CONSOLE PORT SERVER SSH 4-PORT AND CABLES

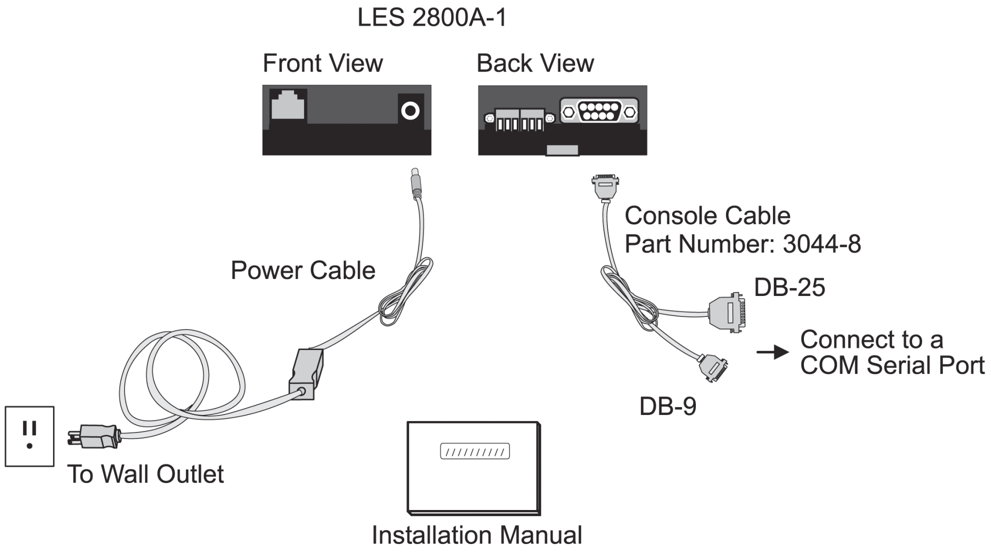


FIGURE 2.6 SECURE CONSOLE PORT SERVER SSH 1-PORT AND CABLES

CHAPTER 4 QUICK INSTALLATION GUIDE

For users familiar with networking, command line interface in Linux or WEB, this chapter gives all the necessary information to quickly configure and start using the Secure Console Port Server SSH box. For more detailed information, the next two chapters should be read.

Configuring using Web

The Secure Console Port Server SSH box comes with an IP address pre-configured on its Ethernet interface (192.168.160.10). To access that box using your browser please do as follows:

Step 1: From the working station, issue a command to add a route pointing to the network 192.168.160.0 reached through the workstation's Ethernet interface.

For Linux, the command would be:

```
      route add -net 192.168.160.0/24 gw <IP address assigned to the workstation's Ethernet interface>
```

e.g. if the workstation has IP address 200.246.93.150 the command would be:

```
      route add -net 192.168.160.0/24 gw 200.246.93.150
```

For Windows, the command would be:

```
      route add 192.168.160.0 mask 255.255.255.0 <IP address assigned to the workstation's Ethernet interface>
```

e.g. if the workstation has IP address 200.246.93.150 the command would be:

```
      route add 192.168.160.0 mask 255.255.255.0 200.246.93.150
```

Step 2: Point your browser to 192.168.160.10

Step 3: Enter **root** as login name and **tslinux** as password

Step 4: Start configuring the parameters presented on the WEB page

☑	WARNING! Type root in the username field and tslinux in the password field to use the Web Configuration Manager. Change the root password as soon as possible: the user database for the Web Configuration Manager is different than the system user database, so the root password can be different.

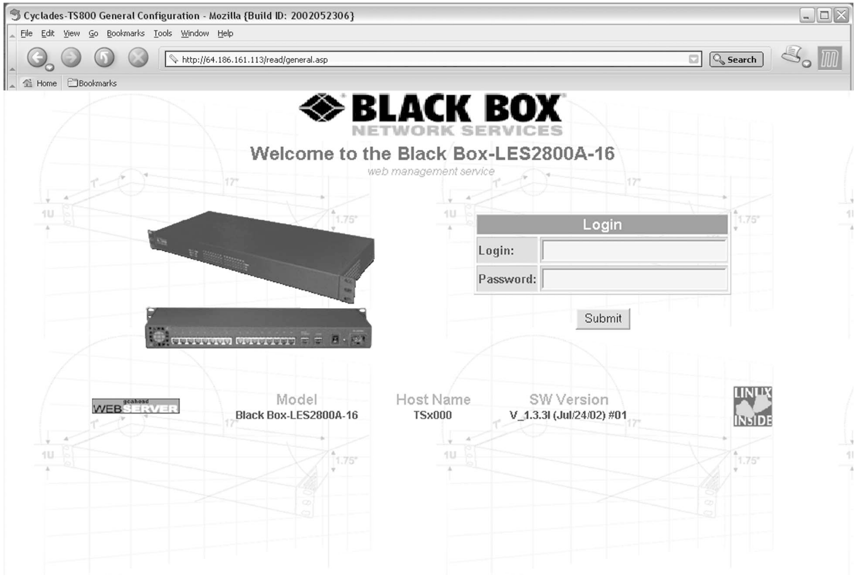
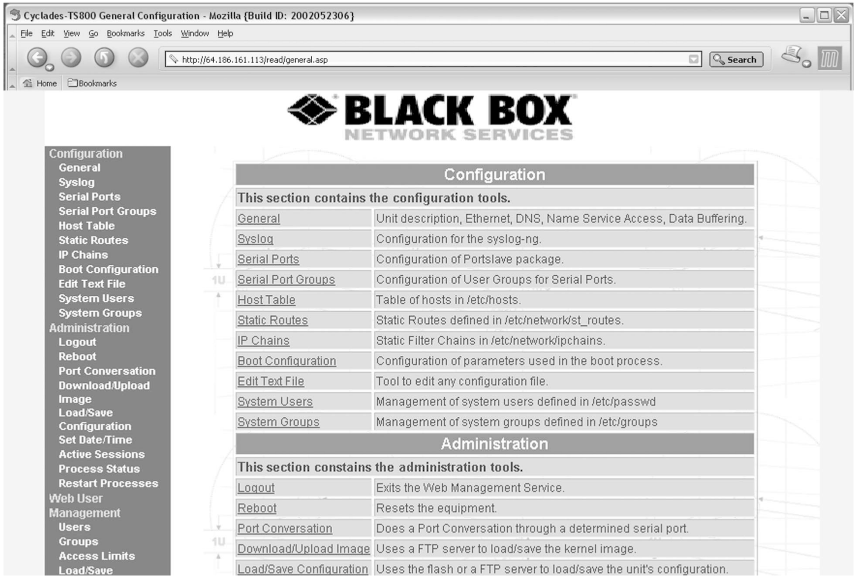

Cyclades-TS800 General Configuration - Mozilla {Build ID: 2002052306}
File Edit View Go Bookmarks Tools Window Help
http://64.186.161.113/read/general.asp
Search
Home
Bookmarks
BLACK BOX
NETWORK SERVICES
Welcome to the Black Box-LES2800A-16
web management service
Login
Login:
Password:
Submit
WEBSERVER
Model
Black Box-LES2800A-16
Host Name
TSx000
SW Version
V_1.3.3l (Jul/24/02) #01
LINUX INSIDE

FIGURE 4.1 LOGIN PAGE OF THE WEB CONFIGURATION MANAGER

After logging in, the screen shown in Figure 4.2 appears.



Configuration	
This section contains the configuration tools.	
General	Unit description, Ethernet, DNS, Name Service Access, Data Buffering.
Syslog	Configuration for the syslog-ng.
Serial Ports	Configuration of Portslave package.
Serial Port Groups	Configuration of User Groups for Serial Ports.
Host Table	Table of hosts in /etc/hosts.
Static Routes	Static Routes defined in /etc/network/st_routes.
IP Chains	Static Filter Chains in /etc/network/ipchains.
Boot Configuration	Configuration of parameters used in the boot process.
Edit Text File	Tool to edit any configuration file.
System Users	Management of system users defined in /etc/passwd
System Groups	Management of system groups defined in /etc/groups

Administration	
This section constains the administration tools.	
Logout	Exits the Web Management Service.
Reboot	Resets the equipment.
Port Conversation	Does a Port Conversation through a determined serial port.
Download/Upload Image	Uses a FTP server to load/save the kernel image.
Load/Save Configuration	Uses the flash or a FTP server to load/save the unit's configuration.

FIGURE 4.2 PAGE FOLLOWING LOGIN

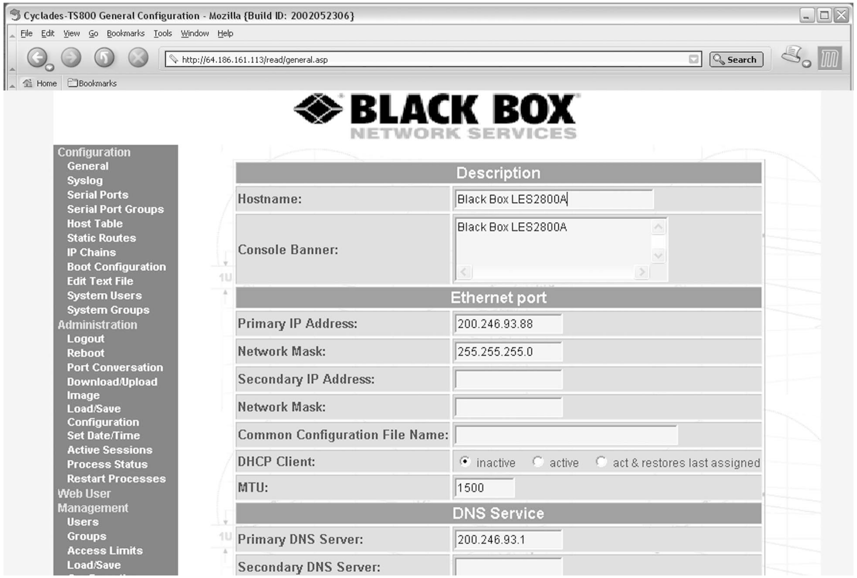
This page gives a brief description of all menu options.

To change the password:

1. Click on the link *Web User Management->Users*
2. Select the user root, then click on the *Change Password* button.
3. Type the new password twice and submit the request.
4. The next page will require a new login, type root and the new password
5. Click on the link *Web User Management->Load/Save Configuration* and click on the *Save Configuration* button.
6. Then, click on the link *Administration->Load/Save Configuration* and click on the *Save Configuration to flash button*.

To logout, click on the *Administration->Log out* link.

The General page of the Web Configuration Manager is shown in Fig. 4.3



Description

Hostname:	Black Box LES2800A
Console Banner:	Black Box LES2800A

Ethernet port

Primary IP Address:	200.246.93.88
Network Mask:	255.255.255.0
Secondary IP Address:	
Network Mask:	
Common Configuration File Name:	
DHCP Client:	[✓ inactive] active act & restores last assigned
MTU:	1500

DNS Service

Primary DNS Server:	200.246.93.1
Secondary DNS Server:	

FIGURE 4.3 GENERAL PAGE OF THE WEB CONFIGURATION MANAGER

A Menu of links is provided along the left side of the page. A summary of what each link leads to is shown in the following figures.

Link Name	**Description of Page Contents**
General	Description, Ethernet, DNS, Name Service Access, Data Buffering.
Syslog	Configuration for the syslog-ng.
Serial Ports	Configuration for the Portslave package.
Serial Port Groups	User Groups in Serial Ports Configuration.
Host Table	Table of hosts in /etc/hosts.
Static Routes	Static routes defined in /etc/network/st_routes.
IP Chains	Static Firewall Chains in /etc/network/ipchains.
Boot Configuration	Configuration of parameters used in the boot process.
Edit Text File	Tool to read and edit a configuration file.
System Users	Management of system users defined in /etc/passwd.
System Groups	Management of system groups defined in /etc/groups.

FIGURE 4.4 THE CONFIGURATION SECTION

Link Name	**Description of Page Contents**
Users	List of users allowed to access the web server.
Groups	List of possible access groups.
Access Limits	List of access limits for specific URL's.
Load/Save Configuration	Load/Save web user configuration in /etc/websum.conf.

FIGURE 4.5 THE WEB USER MANAGEMENT SECTION

Link Name	Description of Page Contents
Logout	Exits the Web Manager.
Reboot	Resets the equipment.
Port Conversation	Does a port conversation through a serial port.
Download/Upload Image	Uses an FTP server to load and save a kernel image.
Load/Save Configuration	Uses flash memory or an FTP server to load or save the system configuration.
Set Date/Time	Set the system date and time.
Active Sessions	Shows the active sessions and allows the administrator to kill them.
Process Status	Shows the running processes and allows the administrator to kill them.
Restart Processes	Allows the administrator to start or stop some processes.

FIGURE 4.6 THE ADMINISTRATION SECTION

Link Name	Description of Page Contents
Interface Statistics	Shows statistics for all active interfaces.
DHCP client	Shows the DHCP client information.
Serial Ports	Shows the status of all serial ports
Routing Table	Shows the routing table and allows the administrator to add or delete routes.
ARP Cache	Shows the ARP cache.
IP Chains	Shows IP Chains Entries.
IP Rules	Shows Firewall, NAT and IP Accounting rules.
IP Statistics	Shows IP protocol statistics.
ICMP Statistics	Shows ICMP protocol statistics.
TCP Statistics	Shows TCP protocol statistics.
UDP Statistics	Shows UDP protocol statistics.
RAM Disk Usage	Shows the File System.
System Information	Shows information about the kernel, Time, CPU and Memory.

FIGURE 4.7 THE INFORMATION SECTION

Configuring using Telnet

The Secure Console Port Server SSH box comes with an IP address pre-configured on its Ethernet interface (192.168.160.10). To access that box using telnet please do as follows:

Step 1: From the working station, issue a command to add a route pointing to the network 192.168.160.0 reached through the workstation's Ethernet interface.

For Linux, the command would be:

```
route add -net 192.168.160.0/24 gw <IP address assigned to the workstation's Ethernet interface>
```

e.g. if the workstation has IP address 200.246.93.150 the command would be:

```
route add -net 192.168.160.0/24 gw 200.246.93.150
```

For Windows, the command would be:

```
route add 192.168.160.0 mask 255.255.255.0 <IP address assigned to the workstation's Ethernet interface>
```

e.g. if the workstation has IP address 200.246.93.150 the command would be:

```
route add 192.168.160.0 mask 255.255.255.0 200.246.93.150
```

Step 2: telnet 192.168.160.10

Step 3: Enter **root** as login name and **tslinux** as password

NOTE: Now, to configure the basic parameters for the Secure Console Port Server SSH, type "wiz" at the command prompt. This will start the wizard configuration application, which can be run at any time from the command prompt. In future firmware releases, more functions will be added to the wizard to allow advanced mode configuration be performed using wizard. (See Appendix F Configuration Wizard for more information).

CHAPTER 5 SUMMARY OF THE CONFIGURATION PROCESS

The Secure Console Port Server SSH can be used as a:

- console server,
- terminal server,
- remote access server.

A detailed description of each of these profiles is provided in the next chapter. The Secure Console Port Server SSH's operating system is embedded Linux. Even if you are a Unix user and find the tools and files familiar, do not configure this product as you would configure a regular Linux server.

You do not need to be a Unix user to configure the Secure Console Port Server SSH. Additional information about the files and tools needed for configuration is provided in appendix A.

The basic configuration steps are:

A. Connecting the Secure Console Port Server SSH to the network and other devices. Consult Chapter 3, What is in the Box, for questions on which cable should be used for which device.

B. Connect a PC or terminal to the Secure Console Port Server SSH via the console port and login.

C. Modify the Linux following files to let the Secure Console Port Server SSH know about its local environment:

```
/etc/hostname
/etc/TIMEZONE (see Appendix K for more information)
/etc/hosts
/etc/resolv.conf
/etc/network/st_routes
```

`/etc/inittab` (Secure Console Port Server SSH 1-Port only. See "Configuring the Secure Console Port Server SSH 1-Port for the First Time" in chapter 6)

D. Change password for root and new users.

The default /etc/passwd file has the user “root” with password “tslinux”. The customer should change the password for user root as soon as possible. Before changing any password or adding new users the customer should also activate shadow password, if it is needed. The Secure Console Port Server SSH has support for shadow password, but it is not active by default. To activate shadow password follow the steps listed below:

1. Create an empty file /etc/shadow

```
 # cd /etc
# touch shadow
```

2. Add a temporary user to the system, it will be removed later.

```
# adduser boo
```

3. Edit the file shadow. For each user in passwd file, create a copy of the line that begins with “boo:” in the shadow file, then replace “boo” with the user name. The root’s line must be the first one.

4. Edit the passwd file and replace all fields password with “x”. The root’s line will look like:

“root:x:0:0:root:/root:/bin/sh”

password field

TIP: Using the vi editor, put the cursor in the first byte after “root:”, then type “ct:x” plus <ESC>.

5. Remove the temporary user boo.

```
# deluser boo
```

6. Change the password for all users and add the new ones needed.

```
# passwd <username>
 or
# adduser <username>
```

7. Edit config_files file and add a line with “/etc/shadow”.

E. Edit the pslave.conf file. This is the main configuration file that concentrates most product parameters and defines the functionality of the Secure Console Port Server SSH. The modifications made to this file will depend on the profile.

F. Activate the changes.

G. Test the configuration to make sure the ports have been set up properly.

H. Save the changes and restart the server application.

Full details on each step listed above and how to perform them are provided in the next chapter. Make sure to always complete ALL the steps for your application before testing or switching to another profile.

☑	WARNING! The Secure Console Port Server SSH provides both a command-line and a web interface for your convenience. Both are enabled by default and both have default passwords. Make sure BOTH default passwords (password is **tslinux**) are changed to avoid unauthorized access to your network. To disable the WEB service, refer to Appendix D.

CHAPTER 6 CONFIGURATION

This chapter guides you step by step through the configuration of the Secure Console Port Server SSH for the three principal applications:

1. Console Server,
2. Terminal Server, and
3. Remote Access Server.

Many steps are common to both, so please read the entire chapter before beginning.

STEP ONE

Connect a PC or terminal to the Secure Console Port Server SSH using the console cable. If using a PC, HyperTerminal can be used in the Windows operating system and Kermit or Minicom in the Unix operating system. The terminal parameters should be set as follows:

- Serial Speed: 9600 bps
- Data Length: 8 bits
- Parity: None
- Stop Bits: 1 stop bit
- Flow Control: none
- Ansi emulation (Note: if your terminal does not have ansi emulation, select vt100; then, on the Secure Console Port Server SSH, log in as root and switch to vt100 by typing “TERM=vt100;export TERM”)

When the Secure Console Port Server SSH boots properly, a login banner will appear.

Log in as root (default password is **tslinux**). A new password should be created as soon as possible. The Secure Console Port Server SSH runs Linux, a Unix-like operating system, and those familiar with the Unix operating system will feel quite at home. A description of the Linux file system and basic commands is given in Appendix A at the end of this manual.

STEP TWO

✓	**Any** configuration change must be saved in flash once validated. To save in flash run saveconf (seen later in this chapter). To validate a configuration run signal_ras hup and check for the ending results (seen later in this chapter).

In this step, four Linux files must be modified to identify the Secure Console Port Server SSH and its neighbors. Then, the boot parameters are configured. The operating system provides the vi editor, which is described in the Linux appendix for the uninitiated. The first file is /etc/hostname. The only entry should be the hostname of the Secure Console Port Server SSH. An example is shown in Figure 6.1.

```
Secure_Console_Port_Server_SSH
```

FIGURE 6.1 CONTENTS OF THE /ETC/HOSTNAME FILE

The second file is /etc/hosts. It should contain the IP address for the Ethernet interface and the same hostname entered in the /etc/hostname file. It may also contain IP addresses and host names for other hosts in the network.

```
200.200.200.1    Secure_Console_Port_Server_SSH
200.200.200.2    RadiusServer
127.0.0.1        localhost
```

FIGURE 6.2 CONTENTS OF THE /ETC/HOSTS FILE

The third file that must be modified is /etc/resolv.conf. It must contain the domain name and nameserver information for the network.

```
domain      mycompany.com
nameserver  200.200.200.2
```

FIGURE 6.3 CONTENTS OF THE /ETC/RESOLV.CONF FILE

The fourth file defines static routes and is called /etc/network/st_routes. In the console server example in Figure 6.5, the PR1000 is the gateway router and thus its IP address is configured in this file to be the default gateway. Other static routes are also configured in this file.

```
route add default gw   200.200.200.5
```

FIGURE 6.4 CONTENTS OF THE /ETC/NETWORK/ST_ROUTES FILE

NOTE: We strongly recommend to use 9600 bps console speed. In case you need to use other speed please check the troubleshooting session.

STEP THREE
This is where the configuration for the three profiles - Console Server, Terminal Server and Remote Access Server diverge. Follow step three for the appropriate profile.

STEP THREE - CONSOLE SERVER

A console server application is shown in Figure 6.5.

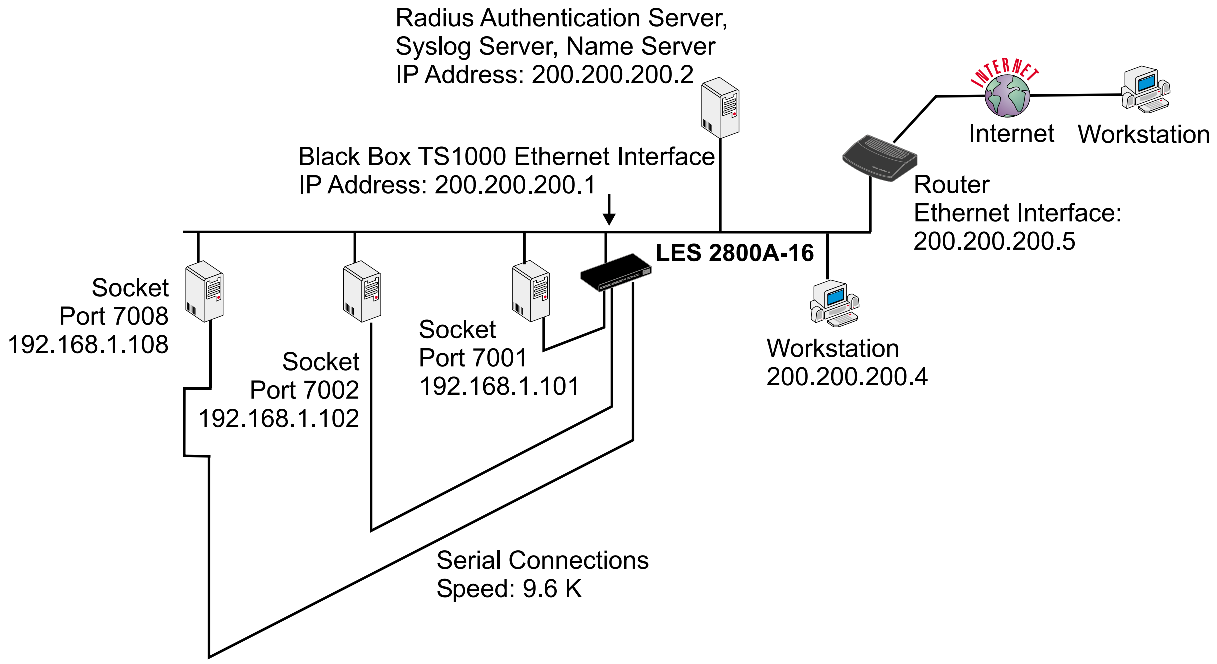


FIGURE 6.5 CONSOLE SERVER APPLICATION

This application allows a user to access a server connected to the Secure Console Port Server SSH through its serial console port from a workstation on the LAN or WAN. A server console is opened on the workstation. The authentication is usually performed by a Radius server and either telnet or ssh (a secure shell session) can be used. See the Linux appendix for more information about ssh.
The fifth file is specific to the Secure Console Port Server SSH and a sample file with comments is supplied in the

Linux file system. It is called /etc/portslave/pslave.conf. A listing of pslave.conf with all possible parameters, as well as sample files used to create the three applications in this chapter, is provided in Appendix C. There are three basic types of parameters: conf.* parameters are global or apply to the Ethernet interface; all.* parameters are used to set default parameters for all ports, and s#.* parameters change the default port parameters for individual ports. An all.* parameter can be overriden by a s#.* parameter appearing later in the pslave.conf file (or vice-versa). A brief description of each parameter used for the console server profile is given in Figures 6.6-6.7.

Parameter	**Description**	**Value for This Example**
conf.eth_ip	The IP address of the Ethernet interface. This parameter, along with the next two, is used by the cy_ras program to OVERWRITE the file /etc/network/ifcfg_eth0 as soon as the command "signal_ras hup" is executed. The file /etc/network/ifcfg_eth0 should not be edited by the user unless the cy_ras application is not going to be used.	200.200.200.1
conf.eth_mask	The mask for the Ethernet network.	255.255.255.0
conf.eth_mtu	The Maximum Transmission Unit size, which determines whether or not packets should be broken up.	1500
conf.nfs_data_ buffering	Remote Network File System where data captured from the serial port will be written instead of the default directory "/var/run/DB". The directory tree to which the file will be written must be NFS-mounted. If data buffering is turned on for port 1, for example, the data will be stored in the file ttyS1.data (or <serverfarm1>.data if s1.serverfarm was configured) in the directory indicated by this variable (please see also Data Buffering section for more details). The remote host must have NFS installed and the administrator must create, export and allow reading/writing to this directory. The size of this file is not limited by the value of the parameter s1.data_buffering, though the value cannot be zero since a zero value turns off data buffering. The size of the file is dependent on the NFS server only (hard drive, partition size, etc.).	commented
conf.lockdir	The lock directory , which is /var/lock for the Secure Console Port Server SSH. It should not be changed unless the user decides to customize the operating system.	/var/lock

FIGURE 6.6 CONSOLE SERVER PSLAVE.CONF GLOBAL PARAMETERS

Parameter	**Description**	**Value for This Example**
conf.facility	This value (0-7) is the Local facility sent to the syslog. The file /etc/syslog-ng/syslog-ng.conf contains a mapping between the facility number and the action (see more in Appendix G).	7
conf.DB_facility	This value (0-7) is the Local facility sent to the syslog with the data when syslog_buffering and/or alarm are active. The file /etc/syslog-ng/syslog-ng.conf contains a mapping between the facility number and the action (see more in Appendix G).	0
conf.group	Used to group users to simplify configuration of the parameter all.users later on. This parameter can be used to define more than one group.	group_name**:** user1, user2

FIGURE 6.6 CONSOLE SERVER PSLAVE.CONF GLOBAL PARAMETERS (CONT.)

Parameter	**Description**	**Value in Exp.**
all.speed	The speed for all ports. . **This value (as for any "all." parameters) can later be overridden for individual ports using the s<*port number*>.speed parameter.**	9600
all.datasize	The data size for all ports.	8
all.stopbits	The number of stop bits for all ports	1
all.parity	The parity for all ports.	none
all.dcd	DCD signal (sets the tty parameter CLOCAL). Valid values are 0 or 1. In a socket session, if all.dcd=0, a connection request (telnet or ssh) will be accepted regardless of the DCD signal and the connection and will not be closed if the DCD signal is set to DOWN. In a socket connection, if all.dcd=1 a connection request will be accepted only if the DCD signal is UP and the connection (telnet or ssh) will be closed if the DCD signal is set to DOWN.	0
all.modbus_ smode	Communication mode through the serial ports. This parameter is meaningful only when modbus protocol is configured. The valid options are ascii (normal TX/RX mode) and rtu (some time constraints are observed between characters while transmitting a frame). If not configured, ASCII mode will be assumed.	commented

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS

Parameter	**Description**	**Value for This Example**
all.authtype	There are several authentication type options: local (authentication is performed using the /etc/passwd file), radius (authentication is performed using a Radius authentication server), TacacsPlus (authentication is performed using a TacacsPlus authentication server), none, local/radius (authentication is performed locally first, switching to Radius if unsuccessful), radius/local (the opposite of the previous option), RadiusDownLocal (local authentication is tried only when the Radius server is down), local/TacacsPlus (authentication is performed locally first, switching to TacacsPlus if unsuccessful), TacacsPlus/local (the opposite of the previous option), TacacsPlusDownLocal (local authentication is tried only when the TacacsPlus server is down). Note that this parameter controls the authentication required by the Secure Console Port Server SSH. The authentication required by the device to which the user is connecting is controlled separately.	radius
all.authhost1	This address indicates the location of the Radius/TacacsPlus authentication server and is only necessary if this option is chosen in the previous parameter. A second Radius/TacacsPlus authentication server can be configured with the parameter all.authhost2.	200.200.200.2
all.accthost1	This address indicates the location of the Radius/TacacsPlus accounting server, which can be used to track how long users are connected after being authorized by the authentication server. Its use is optional. If this parameter is not used, accounting will not be performed. If the same server is used for authentication and accounting, both parameters must be filled with the same address. A second Radius/TacacsPlus accounting server can be configured with the parameter all.accthost2.	200.200.200.2

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS(CONT.)

Parameter	Description	Value for This Example
all.radtimeout	This is the timeout (in seconds) for a Radius/TacacsPlus authentication query to be answered. The first server (authhost1) is tried "radretries" times, and then the second (authhost2), if configured, is contacted "radretries" times. If the second also fails to respond, Radius/TacacsPlus authentication fails.	3
all.radretries	Defines the number of times each Radius/TacacsPlus server is tried before another is contacted. The default, if not configured, is 5.	5
all.secret	This is the shared secret necessary for communication between the Secure Console Port Server SSH and the Radius/TacacsPlus servers.	black box
all.ipno	This is the default IP address of the Secure Console Port Server SSH's serial ports. The "+" indicates that the first port should be addressed as 192.168.1.101 and the following ports should have consecutive values. Any host can access a port using its IP address as long as a path to the address exists in the host's routing table.	192.168.1.101+
all.issue	This text determines the format of the login banner that is issued when a connection is made to the Secure Console Port Server SSH. \n represents a new line and \r represents a carriage return. Expansion characters, listed in Appendix C, can be used here.	\r\n\ TSLINUX - Portslave Internet Services\n\ \r\n\ Welcome to terminal server %h port S%p \n\ \r\n\
all.prompt	This text defines the format of the login prompt. Expansion characters, listed in Appendix C, can be used here.	%h login:
all.flow	This sets the flow control to hardware, software, or none.	hard

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	Description	Value for This Example
all.poll_interval	Valid only for protocols socket_server, socket_ssh and raw_data. When not set to zero, this parameter sets the wait for a TCP connection keep-alive timer (in milliseconds). If no traffic passes through the Secure Console Port Server SSH for this period of time, the Secure Console Port Server SSH will send a line status message to the remote device to see if the connection is still up. If not configured, 1000 ms is assumed. If set to zero, line status messages will not be sent to the socket client.	0
all.socket_port	This defines an alternative labeling system for the Secure Console Port Server SSH ports. The '+' after the numerical value causes the interfaces to be numbered consecutively. In this example, interface 1 is assigned the port value 7001, interface 2 is assigned the port value 7002, etc.	7001+
all.protocol	For the console server profile, the possible protocols are socket_server (when telnet is used), socket_ssh (when ssh version one or two is used), raw_data (to exchange data in transparent mode – similar to socket_server mode, but without telnet negotiation, breaks to serial ports, etc.), or modbus (an application layer messaging protocol for clent/server communication widely used for industrial automation – see Appendix I for details on Modbus protocol).	socket_server

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	Description	Value for This Example
all.data_buffering	A non zero value activates data buffering (local or remote, according to what was configured in the parameter conf.nfs_data_buffering seen before). If local data buffering, a file is created on the Secure Console Port Server SSH; if remote, a file is created through NFS in a remote server. All data received from the port is captured in this file. If local data buffering, this parameter means the maximum file size (in bytes). If remote, this parameter is just a flag to activate (greater than zero) or deactivate data buffering. When local data buffering is used, each time the maximum is reached the oldest 10% of stored data is discarded, releasing space for new data (FIFO system) - circular file. When remote data buffering is used, there's no maximum file size other than the one imposed by the remote server - linear file. This file can be viewed using the normal Unix tools (cat, vi, more, etc.). See the section on data buffering for details.	0
all.DB_timestamp	A non zero value activates time stamp recording in the data buffering file. This parameter is meaningful only if data buffering option is active. In case time stamp recording is on, input characters will be accumulated until either a CR or LF character is received from the serial port or the size of the accumulated data reaches 256 characters. Then, the accumulated data will be recorded in the data buffering file along with the current time.	0
all.syslog_buffering	When non zero, the contents of the data buffer are sent to the syslog-ng every time a quantity of data equal to this parameter is collected. The syslog level for data buffering is hard coded to level 5 (notice) and facility conf.DB_facility. The file /etc/syslog-ng/syslog-ng.conf should be set accordingly for the syslog-ng to take some action (please see Appendix G for syslog-ng configuration file).	0

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	Description	Value for This Example
all.dont_show_DB menu	When zero, a menu with data buffering options is shown when a nonempty data buffering file is found. When 1, the data buffering menu is not shown. When 2, the data buffering menu is not shown but the data buffering file is shown if not empty. When 3, the data buffering menu is shown, but without the **erase** and **show and erase** options.	1
all.alarm	When non zero, all data received from the port are captured and sent to syslog-ng with LOCAL [0+DB_facility] facility and INFO level. The file /etc/syslog-ng/syslog-ng.conf should be set accordingly, for the syslog-ng to take some action (please see Appendix G for syslog-ng configuration file).	0
all.users	Restricts access to ports by user name (only the users listed can access the port or, using the character "!', all but the users listed can access the port .) In this example, the users joe, mark and members of user_group cannot access the port. A single comma and spaces/tabs may be used between names. A comma may not appear between the ! and the first user name. The users may be local, Radius or TacacsPlus. User groups (defined with the parameter conf.group) can be used in combination with user names in the parameter list. Notice that these are common users, not administrators.	! joe, mark, user_group
all.sniff_mode	This parameter determines what other users connected to the very same port (see parameter admin_users below) can see of the session of the first connected user (main session): **in** shows data written to the port, **out** shows data received from the port, and **i/o** shows both streams. The second and later sessions are called sniff sessions and this feature may be activated only when the protocol parameter is set to socket_ssh or socket_server.	out

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	**Description**	**Value for This Example**
all.admin_users	This parameter determines which users can open a *sniff session*, which is where other users connected to the very same port can see everything that a first user connected is doing. The other users connected to the very same port can also cancel the first user's session (and take over). If all.multiple_sessions (seen below) is configured as **no** only two users can connect to the same port simultaneously. If all.multiple_sessions is configured as **yes** more simultaneous users can sniff the session or have read and/or write permission (please see details in Appendix E). When users want access per port to be controlled by administrators, this parameter is obligatory and authtype must not be **none**. This parameter can determine who can open a sniff session or cancel a previous session. User groups (defined with the parameter conf.group) can be used in combination with user names in the parameter list.	peter, john, user_group
all.multiple_sessions	Valid for all serial ports; must be "yes" or "no". If it is not defined, the default will be "no". Please see Appendix E for details.	no
all.escape_char	Valid for all the serial ports with session sniffing enabled (all.admin_users); this parameter will be used to present the menus to the user. The format of this parameter will be set as "^x", where x is the keystroke of the escape character. Only characters from "^a" to "^z" (i.e. CTRL-A to CTRL-Z) will be accepted. The default value is "^z". Please see Appendix E for details.	^z
all.tx_interval	Valid for protocols *socket_server*, *socket_ssh* and *raw_data*. Defines the delay (in milliseconds) before transmission to the Ethernet of data received through a serial port. If not configured, 100ms is assumed. If set to zero or a value above 1000, no buffering will take place.	100

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	**Description**	**Value for This Example**
all.idletimeout	Valid only for the CAS profile (protocols *socket_server*, *socket_ssh raw_data* and *modbus*). Specifies how long (in minutes) a connection can remain inactive before it is cut off. If set to zero (the default), the connection will not time out.	0
all.sttyCmd	Tty settings after a socket connection to that serial port is established. The tty is programmed to work as a CAS profile and this user specific configuration is applied over that serial port. Parameters must be separated by space. (e.g., the following example sets -**igncr** which tells the terminal not to ignore the carriage-return on input, -**onlcr** do not map newline character to a carriage return/newline character sequence on output, **opost** post-process output, -icrnl do not map carriage-return to a newline character on input. all.sttyCmd -igncr -onlcr opost -icrnl)	commented
s1.tty	The device name for the port is set to the value given in this parameter. If a device name is not provided for a port, it will not function.	ttyS1

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Parameter	**Description**	**Value for This Example**
s1.authtype	There are several authentication type options: local (authentication is performed using the /etc/passwd file), radius (authentication is performed using a Radius authentication server), TacacsPlus (authentication is performed using a TacacsPlus authentication server), none, local/radius (authentication is performed locally first, switching to Radius if unsuccessful), radius/local (the opposite of the previous option), RadiusDownLocal (local authentication is tried only when the Radius server is down), local/TacacsPlus (authentication is performed locally first, switching to TacacsPlus if unsuccessful), TacacsPlus/local (the opposite of the previous option), TacacsPlusDownLocal (local authentication is tried only when the TacacsPlus server is down). Note that this parameter controls the authentication required by the Secure Console Port Server SSH. The authentication required by the device to which the user is connecting is controlled separately. Note: if the sniff session feature is used for a specific port, authtype parameter must not be set to none. If none is chosen, any user can open a sniff session and/or cancel sessions of other users for this port.	local
s1.serverfarm	Alias name given to the server connected to the serial port.	Server_connect-ed_serial1
s2.tty	See the s1.tty entry in this table.	ttyS2
s8.tty	See the s1.tty entry in this table.	ttyS8

FIGURE 6.7 CONSOLE SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONT.)

Execute the command **`signal_ras hup`** to activate the changes. At this point, the configuration should be tested. A step-by-step check list follows.

1. Since Radius authentication was chosen, create a new user on the Radius authentication server called test and provide him with the password test.

2. From the console, ping 200.200.200.2 to make sure the Radius authentication server is reachable.
3. Make sure that the physical connection between the Secure Console Port Server SSH and the servers is correct. A cross cable (not the modem cable provided with the product) should be used. Please see the hardware specifications appendix for pin-out diagrams.
4. The Secure Console Port Server SSH has been set for communication at 9600 bps, 8N1. The server must also be configured to communicate on the serial console port with the same parameters. Also make sure that the computer is configured to route console data to its serial console port (Console Redirection).
5. From a server on the LAN (not from the console), try to telnet to the server connected to the first port of the Secure Console Port Server SSH using the following command:

```
telnet 200.200.200.1 7001
```

For both telnet and ssh sessions, the servers can be reached by either:
1. Ethernet IP of the Secure Console Port Server SSH and assigned socket port
or
2. Individual IP assigned to each port.

If everything is configured correctly, a telnet session should open on the server connected to port 1. If not, check the configuration, follow the steps above again, and check the troubleshooting appendix. Now continue on to step four later in this chapter.

NOTE: It is possible to access the serial ports from Microsoft stations using some off-the-shelf packages. Although Black Box is not liable for those packages, successful tests were done using at least one of them. From the application's viewpoint running on a Microsoft station, the remote serial port works like a regular COM port. All the I/O with the serial device attached to the Secure Console Port Server SSH is done through socket connections opened by these packages and a COM port is emulated to the application.

STEP THREE - TERMINAL SERVER

The terminal server profile allows a terminal user to access a server on the LAN. The terminal can be either a dumb terminal or a terminal emulation program on a PC. No authentication is used in this example and rlogin is chosen as the protocol.

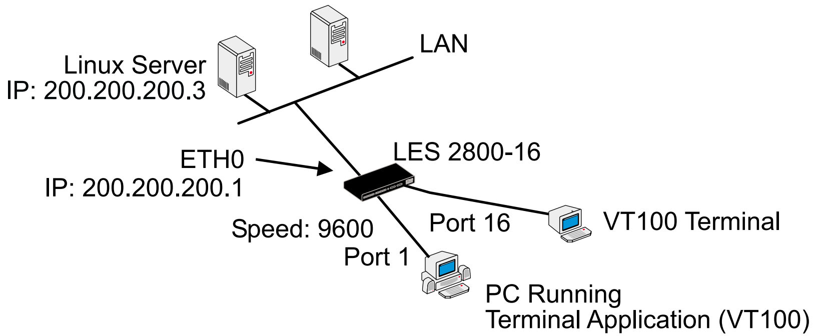


FIGURE 6.8 TERMINAL SERVER APPLICATION

The fifth configuration file (the first four were described in step two) is specific to the Secure Console Port Server SSH and a sample file with comments is supplied in the Linux file system. It is called /etc/portslave/pslave.conf. A listing of pslave.conf with all possible parameters, as well as sample files used to create the three applications in this chapter, is provided in Appendix C. There are three basic types of parameters: conf.* parameters are global or apply to the Ethernet interface; all.* parameters are used to set default parameters for all ports, and s#.* parameters change the default port parameters for individual ports. An all.* parameter can be overriden by a s#.* parameter appearing later in the pslave.conf file (or vice-versa). A brief description of each parameter used for the terminal server profile is given in Figures 6.9-6.10.

Parameter	Description	Value for This Example
conf.eth_ip	The IP address of the Ethernet interface. This parameter, along with the next two, is used by the cy_ras program to OVERWRITE the file /etc/network/ifcfg_eth0 as soon as the command "signal_ras hup" is executed. The file /etc/network/ifcfg_eth0 should not be edited by the user unless the cy_ras application is not going to be used.	200.200.200.1
conf.eth_mask	The mask for the Ethernet network.	255.255.255.0
conf.eth_mtu	The Maximum Transmission Unit size, which determines whether or not packets should be broken up.	1500
conf.lockdir	The lock directory , which is /var/lock for the Secure Console Port Server SSH. It should not be changed unless the user decides to customize the operating system.	/var/lock
conf.rlogin	Location of the rlogin binary that accepts the -i flag.	/usr/local/bin/ rlogin-radius
conf.telnet	Location of the telnet utility.	/bin/telnet
conf.ssh	Location of the ssh utility.	/bin/ssh
conf.locallogins	This parameter is only necessary when authentication is being performed for a port. When set to one, it is possible to log in to the Secure Console Port Server SSH directly by placing a "!" before your login name, then using your normal password. This is useful if the Radius authentication server is down.	0

FIGURE 6.9 TERMINAL SERVER PSLAVE.CONF GLOBAL PARAMETERS

Parameter	Description	Value for This Example
all.speed	The speed for all ports. **This value (as for any "all." parameters) can later be overridden for individual ports using the s<*port number*>.speed parameter.**	9600
all.datasize	The data size for all ports.	8
all.stopbits	The number of stop bits for all ports	1
all.parity	The parity for all ports.	none
all.dcd	DCD signal (sets the tty parameter CLOCAL). Valid values are 0 or 1. In a socket session, if all.dcd=0, a connection request (telnet or ssh) will be accepted regardless of the DCD signal and the connection and will not be closed if the DCD signal is set to DOWN. In a socket connection, if all.dcd=1 a connection request will be accepted only if the DCD signal is UP and the connection (telnet or ssh) will be closed if the DCD signal is set to DOWN.	0
all.authtype	There are several authentication type options: local (authentication is performed using the /etc/passwd file), radius (authentication is performed using a Radius authentication server), TacacsPlus (authentication is performed using a TacacsPlus authentication server), none, local/radius (authentication is performed locally first, switching to Radius if unsuccessful), radius/local (the opposite of the previous option), RadiusDownLocal (local authentication is tried only when the Radius server is down), local/TacacsPlus (authentication is performed locally first, switching to TacacsPlus if unsuccessful), TacacsPlus/local (the opposite of the previous option), TacacsPlusDownLocal (local authentication is tried only when the TacacsPlus server is down). Note that this parameter controls the authentication required by the Secure Console Port Server SSH. The authentication required by the device to which the user is connecting is controlled separately.	none

FIGURE 6.10 TERMINAL SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS

Parameter	**Description**	**Value for This Example**
all.authhost1	This address indicates the location of the Radius/TacacsPlus authentication server and is only necessary if this option is chosen in the previous parameter. A second Radius/TacacsPlus authentication server can be configured with the parameter all.authhost2.	200.200.200.2
all.accthost1	This address indicates the location of the Radius/TacacsPlus accounting server, which can be used to track how long users are connected after being authorized by the authentication server. Its use is optional. If this parameter is not used, accounting will not be performed. If the same server is used for authentication and accounting, both parameters must be filled with the same address. A second Radius/TacacsPlus accounting server can be configured with the parameter all.accthost2.	200.200.200.2
all.radtimeout	This is the timeout (in seconds) for a Radius/TacacsPlus authentication query to be answered. The first server (authhost1) is tried "radretries" times, and then the second (authhost2), if configured, is contacted "radretries" times. If the second also fails to respond, Radius/TacacsPlus authentication fails.	3
all.radretries	Defines the number of times each Radius/TacacsPlus server is tried before another is contacted. The default, if not configured, is 5.	5
all.secret	This is the shared secret necessary for communication between the Secure Console Port Server SSH and the Radius/TacacsPlus servers.	black box
all.protocol	For the terminal server profile, the possible protocols are login (which requests username and password), rlogin (which receives the username from the Secure Console Port Server SSH and requests a password), telnet, socket_client, ssh and ssh2.	rlogin
all.host	The IP address of the host to which the terminals will connect.	200.200.200.3

FIGURE 6.10 TERMINAL SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONTINUED)

Parameter	Description	Value for This Example
all.issue	This text determines the format of the login banner that is issued when a connection is made to the Secure Console Port Server SSH. \n represents a new line and \r represents a carriage return.	\r\n\ TSLINUX - Portslave Internet Services\n\ \r\n\ Welcome to terminal server %h port S%p \n\ \r\n\
all.prompt	This text defines the format of the login prompt. Expansion characters, listed in Appendix C, can be used here.	%h login:
all.term	This parameter defines the terminal type assumed when performing rlogin or telnet to other hosts.	vt100
all.flow	This sets the flow control to hardware, software, or none.	hard
all.socket_port	This parameter defines the port(s) to be used by the protocols telnet and socket_client. If not configured, a default value of 23 is used. Note: socket_server is not valid in this case (TS profile).	23
all.userauto	Username used when connected to a Unix server from the user's serial terminal.	
s1.tty	The device name for the port is set to the value given in this parameter. If a device name is not provided for a port, it will not function.	ttyS1
s16.tty	See the s1.tty entry in this table.	ttyS16

FIGURE 6.10 TERMINAL SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONTINUED)

Execute the command **`signal_ras hup`** to activate the changes. At this point, the configuration should be tested. A step-by-step check list follows.

1. Since authentication was set to none, the Secure Console Port Server SSH will not authenticate the user. However, the Linux Server receiving the connection will. Create a new user on the server called test and provide him with the password test.
2. From the console, ping 200.200.200.3 to make sure the server is reachable.

3. Make sure that the physical connection between the Secure Console Port Server SSH and the terminals is correct. A cross cable (not the modem cable provided with the product) should be used. Please see the hardware specifications appendix for pin-out diagrams.
4. The Secure Console Port Server SSH has been set for communication at 9600 bps, 8N1. The terminals must also be configured with the same parameters.
5. From a terminal connected to the Secure Console Port Server SSH, try to log in to the server using the username and password configured in item one.

Now continue on to step four later in this chapter.

STEP THREE - REMOTE ACCESS SERVER

The remote access server profile allows a modem user to access the LAN. Radius authentication is used in this example and ppp is chosen as the protocol.

☑	WARNING! Remote Access Server functionality was only added to provide a secure and effective means of out-of-band access to servers attached to our product. Use exclusively as a Remote Access Server is not advised.

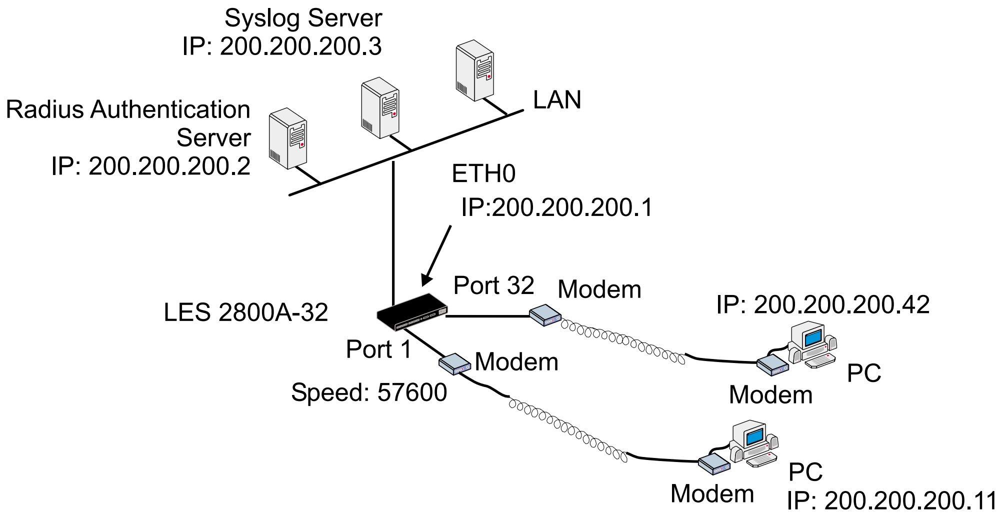


FIGURE 6.11 REMOTE ACCESS SERVER APPLICATION

The fifth configuration file (the first four were described in step two) is specific to the Secure Console Port Server SSH and a sample file with comments is supplied in the Linux file system. It is called /etc/portslave/pslave.conf. A listing of pslave.conf with all possible parameters, as well as sample files used to create the three applications in this chapter, is provided in Appendix C. There are three basic types of parameters: conf.* parameters are global or apply to the Ethernet interface; all.* parameters are used to set default parameters for all ports, and s#.* parameters change the default port parameters for individual ports. An all.* parameter can be overriden by a s#.* parameter appearing later in the pslave.conf file (or vice-versa). A brief description of each parameter used for the remote access server profile is given in Figures 6.12-6.13.

Parameter	**Description**	**Value for This Example**
conf.eth_ip	The IP address of the Ethernet interface. This parameter, along with the next two, is used by the cy_ras program to OVERWRITE the file /etc/network/ifcfg_eth0 as soon as the command "signal_ras hup" is executed. The file /etc/network/ifcfg_eth0 should not be edited by the user unless the cy_ras application is not going to be used.	200.200.200.1
conf.eth_mask	The mask for the Ethernet network.	255.255.255.0
conf.eth_mtu	The Maximum Transmission Unit size, which determines whether or not packets should be broken up.	1500
conf.lockdir	The lock directory , which is /var/lock for the Secure Console Port Server SSH. It should not be changed unless the user decides to customize the operating system.	/var/lock
conf.pppd	Location of the ppp daemon with Radius/TacacsPlus.	/usr/local/sbin/ pppd
conf.facility	This value (0-7) is the Local facility sent to the syslog. The file /etc/syslog-ng/syslog-ng.conf contains a mapping between the facility number and the action (see more in Appendix G).	7

FIGURE 6.12 REMOTE ACCESS SERVER PSLAVE.CONF GLOBAL PARAMETERS

Parameter	Description	Value for This Example
all.speed	The speed for all ports. **This value (as for any "all." parameters) can later be overridden for individual ports using the s<*port number*>.speed parameter.**	57600
all.datasize	The data size for all ports.	8
all.stopbits	The number of stop bits for all ports	1
all.parity	The parity for all ports.	none
all.authtype	There are several authentication type options: local (authentication is performed using the /etc/passwd file), radius (authentication is performed using a Radius authentication server), TacacsPlus (authentication is performed using a TacacsPlus authentication server), none, local/radius (authentication is performed locally first, switching to Radius if unsuccessful), radius/local (the opposite of the previous option), RadiusDownLocal (local authentication is tried only when the Radius server is down), local/TacacsPlus (authentication is performed locally first, switching to TacacsPlus if unsuccessful), TacacsPlus/local (the opposite of the previous option), TacacsPlusDownLocal (local authentication is tried only when the TacacsPlus server is down). Note that this parameter controls the authentication required by the Secure Console Port Server SSH. The authentication required by the device to which the user is connecting is controlled separately.	radius
all.authhost1	This address indicates the location of the Radius/TacacsPlus authentication server and is only necessary if this option is chosen in the previous parameter. A second Radius/TacacsPlus authentication server can be configured with the parameter all.authhost2.	200.200.200.2

FIGURE 6.13 REMOTE ACCESS SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS

Parameter	**Description**	**Value for This Example**
all.accthost1	This address indicates the location of the Radius/TacacsPlus accounting server, which can be used to track how long users are connected after being authorized by the authentication server. Its use is optional. If this parameter is not used, accounting will not be performed. If the same server is used for authentication and accounting, both parameters must be filled with the same address. A second Radius/TacacsPlus accounting server can be configured with the parameter all.accthost2.	200.200.200.2
all.radtimeout	This is the timeout (in seconds) for a radius/TacacsPlus authentication query. The first server (authhost1) is tried "radretries" times, and then the second (if configured) is contacted "radretries" times. If the second also fails to respond, Radius/TacacsPlus authentication fails.	5
all.radretries	Defines the number of times each Radius/TacacsPlus server is tried before another is contacted. The default, if not configured, is 5.	5
all.secret	This is the shared secret necessary for communication between the Secure Console Port Server SSH and the Radius/TacacsPlus servers.	cocomero
all.protocol	For the remote access server profile, the available protocols are PPP, SLIP and CSLIP.	ppp
all.ipno	The IP address to be assigned to the dial-in users. The "+" indicates that the first port should be addressed as 192.168.1.101 and the following ports should have consecutive values.	200.200.200.11+
all.netmask	The netmask corresponding to the IP number provided in the previous parameter.	255.255.255.255
all.mtu	The maximum transmission unit (MTU) that can be transmitted in a PPP packet.	1500
all.mru	The maximum reception unit (MRU) that can be received in a PPP packet.	1500

FIGURE 6.13 REMOTE ACCESS SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONTINUED)

Parameter	Description	Value for This Example
all.initchat	Modem initialization string.	TIMEOUT 10 "" \d\l\dATZ \ OK\r\n-ATZ-OK\r\n "" \ "" ATMO OK\R\N "" \ TIMEOUT 3600 RING "" \ STATUS Incoming %p:I.HANDSHAKE "" ATA \ TIMEOUT 60 CONNECT@ "" \ STATUS Connected %p:I.HANDSHAKE
all.flow	This sets the flow control to hardware, software, or none.	hard
all.autoppp	PPP options to auto-detect a ppp session. The cb-script parameter defines the file used for callback and enables negotiation with the callback server. Callback is available in combination with Radius Server authentication. When a registered user calls the Secure Console Port Server, it will disconnect the user, then call the user back. The following three parameters must be configured in the Radius Server: attribute Service_type(6) : Callback Framed; attribute Framed_Protocol(7): PPP; attribute Callback_Number(19): the dial number (example: 50903300).	%i:%j novj \ proxyarp modem asyncmap 000A0000 \ noipx noccp login auth require-pap refuse-chap \ mtu %t mru %t \ cb-script /etc/portslave/cb_script \ plugin /usr/lib/libpsr.so
all.pppopt	PPP options when user has already been authenticated.	%i:%j novj \ proxyarp modem asyncmap 000A0000 \ noipx noccp mtu %t mru %t netmask %m \ idle %I maxconnect %T \ plugin /usr/lib/libpsr.so

FIGURE 6.13 REMOTE ACCESS SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONTINUED)

Parameter	**Description**	**Value for This Example**
s1.tty	The device name for the port is set to the value given in this parameter. If a device name is not provided for a port, it will not function.	ttyS1
s32.tty	See the s1.tty entry in this table.	ttyS32

FIGURE 6.13 REMOTE ACCESS SERVER PSLAVE.CONF PORT-SPECIFIC PARAMETERS (CONTINUED)

Execute the command **`signal_ras hup`** to activate the changes. At this point, the configuration should be tested. A step-by-step check list follows.

1. Since Radius authentication was chosen, create a new user on the Radius authentication server called test and provide him with the password test.

2. From the console, ping 200.200.200.2 to make sure the Radius authentication server is reachable.

3. Make sure that the physical connection between the Secure Console Port Server SSH and the modems is correct. The modem cable provided with the product should be used. Please see the hardware specifications appendix for pin-out diagrams.

4. The Secure Console Port Server SSH has been set for communication at 57600 bps, 8N1. The modems should be programmed to operate at the same speed on the DTE interface.

5. Try to dial in to the Secure Console Port Server SSH from a remote computer using the username and password configured in item one. The computer dialing in must be configured to receive its IP address from the remote access server (the Secure Console Port Server SSH in this case) and to use PAP authentication.

Now continue on to step four.

STEP FOUR - FOR ALL PROFILES

Secure Console Port Servers SSH 1-Port owners, please skip to the special section on the Secure Console Port Server SSH 1-Port later in this chapter, then return to this section to continue with step four.

Restart the cy_ras process using its process ID. This can be done by executing the command:

```
signal_ras hup
```

This executes the ps command, searches for the cy_ras process id, then sends the signal HUP to the process, all in one step.

Next, the command `saveconf`, which reads the file /etc/config_files, should then be run. The command `saveconf` copies all the files listed in the file /etc/config_files from the ramdisk to /proc/flash/script. The previous contents of the file /proc/flash/script will be lost.

Now the configuration is complete.

`saveconf` is equivalent to `tar -czf /proc/flash/script -T /etc/config_files` in standard Linux (saveconf must be used because tar on the Secure Console Port Server SSH does not support the z flag).

`restoreconf` does the opposite of saveconf, copying the contents of the `/proc/flash/script` file to the corresponding files in the ramdisk. The files on the ramdisk are overwritten. **`restoreconf`** is run automatically each time the Secure Console Port Server is booted.

Information applicable only to the Secure Console Port Server SSH 1-Port

Since there are two physical interfaces available in the Secure Console Port Server SSH 1-Port, RS-232 and RS-485, this model requires the configuration of the parameter described in the Figure 6.14.

Parameter	Description
all.media **or*** s1.media	For the secure Console Port Server 1-Port only. **rs232** (RS-232 interface and DB-9 connector), **rs485_half_terminator** (RS-485 interface, half duplex communication with two wires, DB-9 or block connector, the Secure Console Port Server 1-Port terminates the network), **rs422** (RS-485 interface, full duplex communication with four wires, DB-9 or block connector, the Secure Console Port Server 1-Port terminates the network) or **rs485_half** (RS-485 interface, half duplex communication with two wires, DB-9 or block connector, the Secure Console Port Server 1-Port in the middle of the network).
***NOTE:** all.* parameters are used to set default parameters for all ports and s#.* parameters change the default parameters for individual ports. As this product has only one port, either s1.* or all.* can be used, interchangeably.	

FIGURE 6.14 SECURE CONSOLE PORT SERVER SSH 1-PORT-MEDIA PARAMETER

The next step is to update the system with the modified data in the files above. Make sure the file named /etc/config_files contains the names of all files that should be saved to flash.

Configuring the Secure Console Port Server SSH 1-Port for the first time

The Secure Console Port Server SSH 1-Port does not have a dedicated console port. After configuring the serial port, edit the file /etc/inittab and comment the line that designates the console port (add a “#” to the beginning of the line):

```
# ttyS0::respawn:/sbin/getty -p ttyS0 ansi
```

Next, the command `saveconf`, which reads the /etc/config_files file, should be run. The command `saveconf` copies all the files listed in the file /etc/config_files from the ramdisk to /proc/flash/script. The previous contents of the file /proc/flash/script will be lost.

After rebooting the Secure Console Port Server SSH 1-Port, the initial configuration is complete.

Clustering

Clustering has been added to the Secure Console Port Server SSH with firmware version 1.3.0 (except for the Secure Console Port Server SSH 1-Port). It allows the stringing of Terminal Servers so that one master Secure Console Port Server SSH can be used to access all Secure Console Port Servers SSH on a LAN. The master Secure Console Port Server SSH can manage up to 512 serial ports, so

- 1 Master Secure Console Port Server SSH 16-Port + 31 slave Secure Console Port Servers SSH 16-Port, or
- 1 Master Secure Console Port Server SSH 32-Port + 15 slave Secure Console Port Servers 32-Port, or
- 1 Master Secure Console Port Server SSH 48-Port + 9 slave Secure Console Port Servers 48-Port + 1 slave Secure Console Port Server 32-Port

can be clustered.

An example with one master Secure Console Port Server SSH 32-Port and two slave Secure Console Port Servers SSH 32-Port is shown in Figure 6.15.

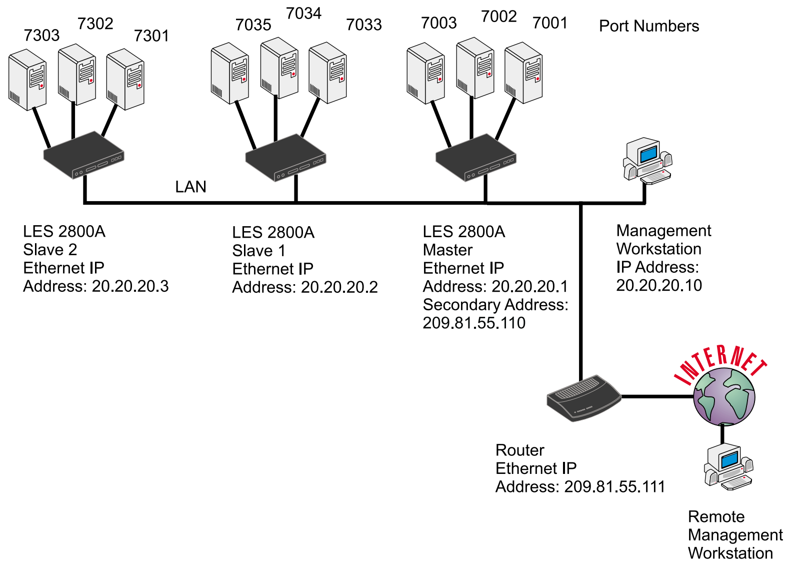


FIGURE 6.16 EXAMPLE USING THE CLUSTERING FEATURE.

The Master Secure Console Port Server SSH must contain references to the Slave ports. The configuration described earlier for Console Access Servers should be followed with the following exceptions for the Master and Slaves:

Master Configuration:

Parameter	Description	Value for This Example
conf.eth_ip	Ethernet Interface IP address.	20.20.20.1
conf.eth_ip_alias	Secondary IP address for the Ethernet Interface (needed for clustering feature).	209.81.55.110
conf.eth_mask_alias	Mask for secondary IP address above.	255.255.255.0
all.socket_port	This value applies to both the local ports and ports on slave Secure Console Port Servers SSH.	7001+
all.protocol	Depends on the application.	Socket_ssh or socket_server
all.authtype	Depends on the application.	Radius or local or none
s33.tty	This parameter must be created in the master Secure Console Port Server SSH file for every slave port. Its format is IP_of_Slave:[slave_socket_port] for non-master ports. In this case, the slave_socket_port value is not necessary because s33.socket_port is automatically set to 7033 by all.socket_port above.	20.20.20.2:7033
s33.serverfarm	An alias for this port.	Server_on_slave1_serial_s1
s33.ipno	This parameter must be created in the master Secure Console Port Server SSH file for every slave port, unless configured using all.ipno.	0.0.0.0
s34.tty	See s33.tty.	20.20.20.2:7034
s34.serverfarm	An alias for this port.	Server_on_slave1_serial_s2
s34.ipno	See s33.ipno.	0.0.0.0

FIGURE 6.16 MASTER SECURE CONSOLE PORT SERVER SSH CONFIGURATION (WHERE IT DIFFERS FROM THE STANDARD CAS PROFILE)

Parameter	Description	Value for This Example
s35.tty	See s33.tty.	20.20.20.2:7035
s35.serverfarm	An alias for this port.	Server_on_slave1_serial_s3
s35.ipno	See s33.ipno.	0.0.0.0
etc. for s36-s64		
S65.tty	The format of this parameter is IP_of_Slave:[slave_socket_port] for non-master ports. The value 7301 was chosen arbitrarily for this example.	20.20.20.3:7301
S65.serverfarm	An alias for this port.	Server_on_slave2_serial_s1
S65.ipno	See s33.ipno.	0.0.0.0
S66.tty	See s65.tty.	20.20.20.3:7302
S66.serverfarm	An alias for this port.	Server_on_slave2_serial_s2
S66.ipno	See s33.ipno.	0.0.0.0
S67.tty	See s65.tty.	20.20.20.3:7303
S67.serverfarm	An alias for this port.	Server_on_slave2_serial_s3
S67.ipno	See s33.ipno.	0.0.0.0
etc. for s68-s96		

FIGURE 6.16 MASTER SECURE CONSOLE PORT SERVER SSH CONFIGURATION (CONT.)

The Slave Secure Console Port Servers SSH do not need to know they are being accessed through the Master Secure Console Port Server SSH. Their port numbers, however, must agree with those assigned by the Master.

Parameter	Value for This Example
all.protocol	socket_server
all.authtype	none
conf.eth_ip	20.20.20.2
all.socket_port	7033+

FIGURE 6.17 SECURE CONSOLE PORT SERVER SSH CONFIGURATION FOR SLAVE 1 (WHERE IT DIFFERS FROM THE STANDARD CAS PROFILE)

Parameter	Value for This Example
all.protocol	Socket_server
all.authtype	None
conf.eth_ip	20.20.20.3
all.socket_port	7301+

FIGURE 6.18 SECURE CONSOLE PORT SERVER SSH CONFIGURATION FOR SLAVE 2 (WHERE IT DIFFERS FROM THE STANDARD CAS PROFILE)

To access ports from the remote management workstation, use telnet with the secondary IP address:

```
Telnet 209.81.55.110 7001 to access the first port of the Master Secure
Console Port Server SSH
Telnet 209.81.55.110 7033 to access the first port of Slave 1
Telnet 209.81.55.110 7065 to access the first port of Slave 2
```

Note that socket port 7065 is being used in the last example to access port 7301 in Slave 2.

ssh can also be used from the remote management workstation:

```
ssh -l <username>:Server_on_slave2_serial_s3 209.81.55.110 to access the
third port of Slave 2
ssh -l <username>:7069 209.81.55.110 to access the fifth port of Slave 2
```

Centralized Management - Include File

The Secure Console Port Server SSH allows centralized management through the use of a master pslave.conf file. Administrator's should consider this approach to configure multiple Secure Console Port Servers SSH. Using this feature, each unit has a simplified pslave.conf file where a master include file is cited. This common configuration file contains information for all units, properly separated in separate sections, and would be stored on one central server. This file, in our example shown in figure 6.19, is /etc/portslave/TScommon.conf. It must be downloaded to each Secure Console Port Server SSH.

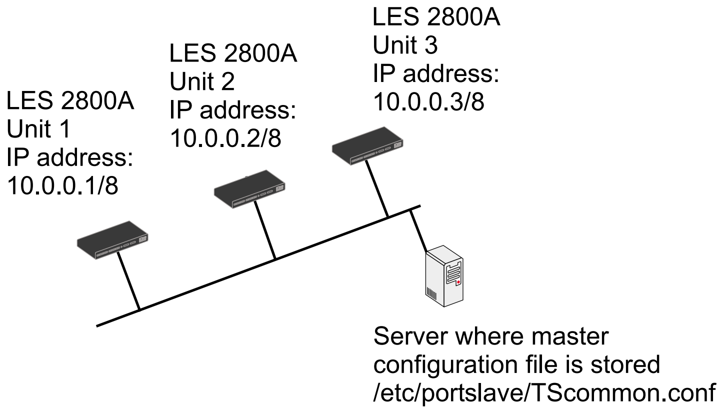


FIGURE 6.19 EXAMPLE OF CENTRALIZED MANAGEMENT

The abbreviated pslave.conf and /etc/hostname files in each unit, for the example are:
unit 1:

```
unit1
```

FIGURE 6.20 /ETC/HOSTNAME FILE IN UNIT 1

```
conf.eth_ip         10.0.0.1
conf.eth_mask       255.0.0.0
conf.include        /etc/portslave/TScommon.conf
```

FIGURE 6.21 PSLAVE.CONF FILE IN UNIT 1

unit 2:

```
unit2
```

FIGURE 6.22 /ETC/HOSTNAME FILE IN UNIT 2

```
conf.eth_ip         10.0.0.2
conf.eth_mask       255.0.0.0
conf.include        /etc/portslave/TScommon.conf
```

FIGURE 6.23 PSLAVE.CONF FILE IN UNIT 2

unit 3:

```
unit3
```

FIGURE 6.24 /ETC/HOSTNAME FILE IN UNIT 1

```
conf.eth_ip         10.0.0.3
conf.eth_mask       255.0.0.0
conf.include        /etc/portslave/TScommon.conf
```

FIGURE 6.25 PSLAVE.CONF FILE IN UNIT 3

The common include file for the example is:

```
conf.host_config  unit1

<parameters for unit1 following the rules for pslave.conf>

conf.host_config  unit2

<parameters for unit2 following the rules for pslave.conf>

conf.host_config  unit3

<parameters for unit3 following the rules for pslave.conf>
conf.host_config .end
```

FIGURE 6.26 TSCOMMON.CONF FILE

When this file is included, unit1 would read only the information between "conf.host_config unit1" and conf.host_config unit2". Unit2 would use only the information between "conf.host_config unit2" and conf.host_config unit3" and unit3 would use information after "conf.host_config unit3" and before conf.host_config .end.

The following steps should be followed to use centralized configuration

1. Create and save the /etc/portslave/pslave.conf and /etc/hostname files in each Secure Console Port Server
2. Execute the command signal_ras hup on each unit.
3. Create and save the common configuration file on the server, then download it (probably using scp) to each unit. Make sure to put it in the directory set in the pslave.conf file (/etc/portslave in the example).
4. Execute the command signal_ras hup on each unit again.
5. Test each unit. If everything works, add the line /etc/portslave/TScommon.conf to the /etc/config_files file. Save the file and close it. Next, execute the saveconf command.

NOTE: The included file /etc/portslave/TScommon.conf cannot contain another include file (i.e. the parameter conf.include must not be defined).

CHAPTER 7 UPGRADES AND TROUBLESHOOTING

Upgrades

All 6 files added by Black Box to the standard Linux files are in the /proc/flash directory. They are:
boot_ori - original boot code
boot_alt - alternate boot code
syslog - event logs (not used by Linux)
config - configuration parameters, only the boot parameters are used by the boot code
zImage - Linux kernel image
script - file where all Secure Console Port Server SSH configuration information is stored

To upgrade the Secure Console Port Server SSH, proceed as follows:

A) Log in to the Secure COnsole Port Server SSH as root (provide the root password if requested)

B) Go to the /proc/flash directory using the following command:

```
cd /proc/flash
```

C) Ftp to the host where the new firmware is located, log in using your username and password, go to the directory where the firmware is located, select binary transfer and “get” the firmware file. NOTE: the destination file name in the /proc/flash directory must be zImage. Example (hostname = server; directory = /tftpboot; username = admin; password = adminpw; firmware filename on that server = zImage.133):

```
ftp
> open server
> user admin
> Password: adminpw
> cd /tftpboot
> bin
> get zImage.133 zImage
> quit
```

NOTE: Due to space limitations, the new zImage file may not be downloaded with a different name, then renamed. The Secure Console Port Server SSH searches for a file named zImage when booting and there is no room in flash for two zImage files.

To make sure the downloaded file is not corrupted or that the zImage saved in flash is OK, run:

```
md5sum -b /proc/flash/zImage
```

Now check with the information present in the text file saved in the Black Box site (e.g. zImage.133.md5sum). If the numbers match the downloaded file is not corrupted.

D) Issue the command reboot

```
reboot
```

E) After rebooting, the new Linux kernel will take over. This can be confirmed by typing `cat /proc/version` to see the Linux kernel version.

Troubleshooting

If the contents of flash memory are lost after an upgrade, please follow the instructions below to restore your system:

a. Turn the Secure Console Port Server SSH OFF, then back ON
b. Using the console, during self test, press <Esc> after the Ethernet test
c. When the Watch Dog Timer prompt appears, press <Enter>
d. Choose the option Network Boot when asked
e. Enter the IP address of the Ethernet interface
f. Enter the IP address of the host where the new zImage file is located
g. Enter the file name of the zImage file on the host
h. Select the TFTP option instead of BOOTP (the host must be running TFTPD and the new zImage file must be located in the proper directory. e.g. /tftpboot for Linux).
i. Accept the default MAC address by pressing <Enter>

j. The Secure Console Port Server SSH should begin to boot off the network and the new image will be downloaded and begin running in RAM. At this point, follow the upgrade steps above (login, cd /proc/flash, ftp, and so forth) to save the new zImage file into flash again.

NOTE: possible causes for the loss of flash memory: downloaded wrong zImage file, downloaded as ASCII instead of binary; problems with flash memory.

If the Secure Console Port Server SSH booted properly, the interfaces can be verified using ifconfig and ping. If ping does not work, check the routing table using the command route. Of course, all this should be tried after checking that the cables are connected correctly.

As mentioned in Chapter 6, the file /etc/config_files contains a list of files acted upon by saveconf and restoreconf. If a file is missing, it will not be loaded onto the ramdisk on boot. The following table lists files that should be included in the /etc/config_files file and which programs use each.

File	**Program**
/etc/securetty	telnet, login, su
/etc/issue	getty
/etc/getty_ttyS0	login (via console)
/etc/hostname	tcp
/etc/hosts	tcp
/etc/host.conf	tcp
/etc/nsswitch.conf	dns
/etc/resolv.conf	dns
/etc/config_files	saveconf
/etc/passwd	login, passwd, adduser...
/etc/group	login, passwd, adduser...
/etc/ssh/ssh_host_key.pub	sshd
/etc/ssh/sshd_config	sshd
/etc/ssh/ssh_config	ssh client
/etc/ssh/ssh_host_key	sshd (ssh1)
/etc/ssh/ssh_host_key.pub	sshd (ssh1)

File	Program
/etc/ssh/ssh_host_dsa_key	sshd (ssh2)
/etc/ssh/ssh_host_dsa_key.pub	sshd (ssh2)
/etc/snmp/snmpd.conf	snmpd
/etc/portslave/pslave.conf	cy_ras, portslave, configuration information
/etc/network/ifcfg_eth0	ifconfig eth0, cy_ras, rc.sysinit
/etc/network/ifcfg*	ifconfig, cy_ras, rc.sysinit
/etc/network/ifcfg_lo	ifconfig lo, cy_ras, rc.sysinit
/var/run/radsession.id	radinit, radius authentication process
/home	adduser, passwd
/etc/network/st_routes	ifconfig, cy_ras, rc.sysinit
/etc/syslog-ng/syslog-ng.conf	syslog-ng

If any of the files listed in /etc/config_files is modified, the Secure Console Port Server SSH administrator must execute the command saveconf before rebooting the Secure Console Port Server SSH or the changes will be lost. If a file is created (or a file name altered), its name must be added to this file before executing saveconf and reboot.

Black Box Support is always ready to help with any configuration problems. Before calling, execute the command

```
cat /proc/version
```

and note the Linux version and Secure Console Port Server SSH version written to the screen. This will speed resolution of most problems.

Hardware Test

A hardware test called tstest is included with the Secure Console Port Server SSH firmware. It is a menu-driven program, run by typing tstest at the command prompt, and the various options are described below.

Note: The Secure Console Port Server SSH should not be tested while in use. The user should inactivate all processes that may use the serial ports. They are inetd, sshd, cy_ras, and cy_buffering. The user should follow the steps below:

step 1) signal_ras stop.

step 2) Perform all hardware tests needed.

step 3) signal_ras start.

Port Test

Either a cross cable or a loop-back connector is necessary for this test. Their pinout diagrams are supplied in appendix B. Connect the loop-back connector to the modem cable and then connect the modem cable to the port to be tested (or connect a cross cable between two ports to be tested). In the case of the Secure Console Port Server SSH 1-Port, connect the DB-25 loop-back connector to the console cable using a DB-9 - DB-25 convertor. When tstest senses the presence of the cable or connector, the test will be run automatically and the result shown on the screen.

Each line of data correponds to a port in test. The last 4 columns (DATA, CTS, DCD, and DSR) indicate errors. The values in these columns should be zero. The figure below is an example of the output screen.

```
                      <- Packets ->                        <- Errors ->
From        To   Sent    Received     Passes    Data    CTS     DCD     DSR
2      <->  2    35      35           35        0       0       0       0
4      <->  5    35      35           35        0       0       0       0
5      <->  4    35      35           35        0       0       0       0
```

When this test is run with a cable or connector without the DSR signal (see the pinout diagram for the cable or connector being used), errors will appear in the DSR column. This does not indicate a problem with the port. In the example above, tstest perceived that a loop-back connector was attached to port 2 and that a cross cable was used to connect ports 4 and 5.

Port Conversation

This test sends and receives data on the selected port. One way to run this test is to place a loop-back connector on the port to be tested and begin. Enter the number of the port and a baud rate (9600 is a typical value). Type some letters, and if the letters appear on the screen, the port is working. If the letters do not appear on the screen (which also occurs if the loop-back connector is removed), the port is not functioning correctly.

A second method that can be used to test the port is to connect it to a modem with a straight cable. Begin the test and type “at”. The modem should respond with “OK”, which will appear on the screen. Other commands can be sent to the modem or to any other serial device.

Test Signals Manually

This test confirms that signals are being sent and received on the selected port. Neither the loop-back connector nor the cross cable are necessary. Enter the number of the port to be tested and begin the test.

```
State    DTR    DCD    DSR    RTS    CTS
-----------------------------------------
ON        X                    X
          ↓                    ↓
OFF              X      X             X
```

First, type Ctrl-D to see the X in the DTR column move position, then type Ctrl-R to see the X in the RTS column change position. If each of the Xs moves in response to its command, the signals are being sent.

Another method to test the signals is to use a loop-back connector. Enter the number of the port with the loop-back connector and start the test. In this case, when Ctrl-D is typed, the Xs in the first three columns will move as shown below.

State	DTR	DCD	DSR	RTS	CTS
ON	X	X	X	X	X
	↓	↓	↓	↓	↓
OFF					

This is because the test is receiving the DTR signal sent through the DCD and DSR pins. When Ctrl-R is typed, the Xs in the RTS and CTS columns should move together. If the Xs change position as described, the signals are being sent and received correctly.

Single User Mode

The Secure Console Port Server SSH has a single user mode used when:

- The name or password of the user with root privileges is lost or forgotten,
- After an upgrade or downgrade which leaves the Secure Console Port Server SSH unstable,
- After a configuration change which leaves the Secure Console Port Server SSH inoperative or unstable.

Type the word “ single” (with a blank space before the word) during boot using a console connection. This cannot be done using a telnet or other remote connection.

The initial output of the boot process is shown below.

```
Entry Point = 0x00002120
loaded at: 00002120 0000D370
relocated to: 00300020 0030B270
board data at: 003052C8 0030537C
relocated to: 002FF120 002FF1D4
zimage at: 00008100 0006827E
relocated to: 00DB7000 00E1717E
initrd at: 0006827E 0024F814
relocated to: 00E18000 00FFF596
avail ram: 0030B270 00E18000
Linux/PPC load: root=/dev/ram
```

After printing “Linux/PPC load: root=/dev/ram”, the Secure Console Port Server SSH waits approximately 10 seconds for user input. This is where the user should type “<sp>single” (spacebar, then the word “single”). When the boot process is complete, the Linux prompt will appear on the console:

```
[root@(none) /]#
```

If the password or username was forgotten, execute the following commands:

```
passwd
saveconf
reboot
```

For configuration problems, the user has two options:

1. Edit the file(s) causing the problem with vi, then execute the commands
```
saveconf
reboot
```

2. Reset the configuration by executing the commands:
```
echo 0 > /proc/flash/script
reboot
```

If the problem is due to an upgrade/downgrade, a second downgrade/upgrade will be necessary to reverse the process. First, the network must be initialized in order to reach a ftp server. Execute the following script, replacing the parameters with values appropriate for your system. If your ftp server is on the same network as the Secure Console Port Server SSH, the gw and mask parameters are optional.

```
config_eth0 ip 200.200.200.1 mask 255.255.255.0 gw 200.200.200.5
```

At this point, the DNS configuration (in the file /etc/resolv.conf) should be checked. Then, download the kernel image using the ftp command.

Troubleshooting the Web Configuration Manager

1. What to do when the initial web page does not appear.

Try pinging, telnetting or tracerouting to the Secure Console Port Server SSH to make sure it is reachable. If not, the problem is probably in the network or network configuration. Are the interfaces up? Are the IP addresses correct? Are filters configured which block the packets?

If the Secure Console Port Server SSH is reachable, see if the /bin/webs process is running by executing the command ps. If it is not, type /bin/webs & to start it. If the /bin/webs process is not being initialized during boot, change the file
/etc/inittab.

2. How to restore the default configuration of the Web Configuration Manager

This would be required only when the root password was lost or the configuration file /etc/websum.conf was damaged.

From a console or telnet session, edit the file /etc/config_files. Find the reference to /etc/websum.conf and delete it. Save the modified /etc/config_files file. Execute the command saveconf. Reboot the system. Enter into the Web Configuration Manager with the default username and password (root/tslinux). Edit the file /etc/ config_files and insert the reference to /etc/websum.conf.

Recover the access to the Secure Console Port Server SSH 1-Port console port

There is no dedicated console port available in the Secure Console Port Server SSH 1-Port. As factory default the serial port is set to work as a console port to allow initial product configuration. After that, changes can still be made through the Ethernet port and a Telnet command. If for some reason this access is lost (usually misconfiguration), the product can only be configured if the steps bellow are followed.

1. Power the Secure Console Port Server SSH 1-Port off.
2. Connect the Secure Console Port Server SSH 1-Port to a terminal configured to work at 9600 bps, with 8 bits, no parity and 1 stop bit.
3. Press and hold the reset button and power on the Secure Console Port Server SSH 1-Port. Release the reset button when the self test starts on the terminal’s screen.

The Secure Console Port Server SSH 1-Port will be now in single user mode, the serial port will work as a console port and the product can de reconfigured. Notice that no previous configuration is lost. After finishing, **save the configuration** (saveconf), power the Secure Console Port Server SSH 1-Port off, and reconnect the original device to the serial port.Using a different speed for the Serial Console.

Using a different speed for the serial console

The serial console is originally configured to work at 9600 bps. If the customer wants to change that, it is necessary to run bootconf. The user will be presented with the screen:

```
Current configuration

MAC address assigned to Ethernet [00:60:2e:00:16:b9]
IP address assigned to Ethernet interface [192.168.160.10]
Watchdog timer ((A)ctive or (I)nactive) [A]
Firmware boot from ((F)lash or (N)etwork) [F]
Boot type ((B)ootp,(T)ftp or Bot(H)) [T]
Boot File Name [zvmppcts.bin]
Server's IP address [192.168.160.1]
Console speed [9600]
(P)erform or (S)kip Flash test [P]
(S)kip, (Q)uick or (F)ull RAM test [F]
Fast Ethernet ((A)uto Neg, (1)00 BtH, 100 Bt(F), 10 B(t)F, 10 Bt(H)) [A]
Fast Ethernet Maximun Interrupt Events [0]
```

Type <Enter> for all fields but the Console Speed. When presented the following line:

Do you confirm these changes in flash ((Y)es, (N)o (Q)uit) [N] :

Enter Y and the changes will be saved in flash. Reboot the unit to have the changes effective and use the console at the new speed.

APPENDIX A INFORMATION FOR USERS NOT FAMILIAR WITH LINUX

Users and Passwords

A username and password are necessary to log in to the Secure Console Port Server SSH. The user “root” is predefined, with a password **tslinux**. A password should be configured as soon as possible to avoid unauthorized access.
Type the command:

```
passwd
```

to create a password for the root user.

To create a regular user (without root privileges), use the commands:

```
adduser user_name
passwd user_name
```

NOTE: If you do not use a combination of upper and lower case letters and numbers, and between 5 and 8 characters for the password, you will get a warning, but it will still be accepted.

To log out, type “logout” at the command prompt.

Linux File Structure

The Linux file system is organized hierarchically, with the base (or root) directory represented by the symbol “/”. All folders and files are nested within each other below this base directory. The directories located just below the base directory are:

/home	Contains the work directories of system users.
/bin	Contains applications and utilities used during system initialization.
/dev	Contains files for devices and ports.
/etc	Contains configuration files specific to the operating system.
/lib	Contains shared libraries.
/proc	Contains process information
/mnt	Contains information about mounted disks.
/opt	Location where packages not supplied with the operating system are stored.
/tmp	Location where temporary files are stored.
/usr	Contains most of the operating system files.
/var	Contains operating system data files.

Basic File Manipulation Commands

The basic file manipulation commands allow the user to copy, delete and move files and create and delete directories.

cp *file_name destination* a) cp text.txt /tmp b) cp /chap/robo.php ./excess.php	Copies the file indicated by *file_name* to the path indicated by *destination*. a) copies the file text.txt in the current directory to the tmp directory. b) copies the file robo.php in the chap directory to the current directory and renames the copy excess.php.
rm *file_name*	Removes the file indicated by *file_name*.
mv *file_name destination*	Moves the file indicated by *file_name* to the path indicated by destination.
mkdir *directory_name* a) mkdir spot b) mkdir /tmp/snuggles	Creates a directory named directory_name. a) creates the directory spot in the current directory. b) creates the directory snuggles in the directory tmp.
rmdir *directory_name*	Removes the directory indicated by *directory_name*.

Other commands allow the user to change directories and see the contents of a directory.

pwd	Supplies the name of the current directory. While logged in, the user is always "in" a directory. The default initial directory is the user's home directory, /home/<username>
ls [options] *directory_name*	Lists the files and directories within *directory_name*. Some useful options are -l for more detailed output and -a which shows hidden system files.
cd *directory_name*	Changes the directory to the one specified
cat *file_name*	Prints the contents of *file_name* to the screen.

Shortcuts:

. (a dot)	represents the current directory
.. (two dots)	represents one directory above the current directory (i.e. one directory closer to the base directory).

The vi Editor

To edit a file using the vi editor, type

```
vi file_name
```

vi is a three-state line editor: it has a command mode, a line mode and an editing mode. If in doubt as to which mode you are in, press the <ESC> key which will bring you to the command mode.

Mode	**What is done there**	**How to Get There**
command mode	navigation within the open file	Press the <ESC> key.
editing mode	text editing	See list of editing commands below.
line mode	file saving, opening, etc. exiting from vi	From the command mode, type ":" (the colon).

Entering the program, the user is automatically in the command mode. To navigate to the part of the file to be edited, use the following keys:

`h`	moves the cursor to the left (left arrow)
`j`	moves the cursor to the next line (down arrow)
`k`	moves the cursor to the previous line (up arrow)
`l`	moves the cursor to the right (right arrow)

Having arrived at the location where text should be changed, use these commands to modify the text (note commands “i” and “o” will move you into the editing mode and everything typed will be taken literally until you press the <ESC> key to return to the command mode)

`i`	insert text before the cursor position (everything to the right of the cursor is shifted right)
`o`	create a new line below the current line and insert text (all lines are shifted down)
`dd`	remove the entire current line
`u`	undo the last modification
`x`	delete the letter at the cursor position

Now that the file has been modified, enter the line mode (by typing “:” from the command mode) and use one of the following commands:

`w`	save the file (w is for write)
`wq`	save and close the file (q is for quit)
`q!`	close the file without saving
`w` *`file`*	save the file with the name *file*
`e` *`file`*	opens the file named *file*

The Routing Table

The Secure Console Port Server SSH has a static routing table that can be seen using the commands

```
     route
```

or

```
     netstat -rn
```

The file /etc/network/st_routes shown in Figure 6.5 is the Secure Console Port Server SSH's method for configuring static routes. Routes should be added to the file (which is a script run when the Secure Console Port Server SSH is initialized) or at the prompt (for temporary routes) using the following syntax:

```
     route [add|del] [-net|-host] target netmask nt_msk [gw gt_way] interf
```

`[add|del]`	one of these tags must be present -- routes can be either added or deleted.
`[-net|-host]`	-net is for routes to a network and -host is for routes to a single host.
`target`	*`target`* is the IP address of the destination host or network
`netmask` *`nt_msk`*	the tag netmask and a mask are necessary only when subnetting is used. Otherwise, a mask appropriate to the *`target`* is assumed. *`nt_msk`* must be specified in dot notation.
`gw` *`gt_way`*	specifies a gateway, when applicable. *`gt_way`* is the IP address or hostname of the gateway.
`interf`	the interface to use for this route. Must be specified if a gateway is not. When a gateway is specified, the operating system determines which interface is to be used.

ssh - The Secure Shell Session

ssh is a command interface and protocol often used by network administrators to connect securely to a remote computer. ssh replaces its non-secure counterpart rsh and rlogin. There are two versions of the protocol, ssh and ssh2. The Secure Console Port Server SSH offers both.

The command to start an ssh client session from a **Unix** workstation is

```
     ssh -t <user>@<hostname>
```

where

```
        <user> = <username>:ttySnn or
                 <username>:socket_port or
                 <username>:ip_addr or
                 <username>:serverfarm
```

Note: “serverfarm” is a physical port alias. It can be configured in the file pslave.conf.

An example:

username:	blackbox
Secure Console Port Server SSH IP address:	192.168.160.1
host name:	scps
servername for port 1:	file_server

ttyS1 addressed by IP 10.0.0.1 or socket port 7001. The various ways to access the server connected to the port are:

```
     ssh -t blackbox:ttyS1@scps
     ssh -t blackbox:7001@scps
     ssh -t blackbox:10.0.0.1@scps
     ssh -t blackbox:file_server@scps
     ssh -t -l blackbox:10.0.0.1
     ssh -t -l blackbox:7001 scps
```

Note that either -l or @ are used, but not both. For openssh version 3.1p1 or later (Secure Console Port Server SSH V_1.3.2 or later), ssh2 is the default. In that case, the -1 flag is used for ssh1.

`ssh -t blackbox:7001@scps` (openssh earlier than 3.1p1 - Secure Console Port Server SSH V_1.3.1 and earlier -> ssh1 will be used)

`ssh -t -2 blackbox:7001@scps` (openssh earlier than 3.1p1 - Secure Console Port Sever SSH V_1.3.1

and earlier -> ssh2 will be used)
`ssh -t blackbox:7001@scps` (openssh 3.1p1 or later - Secure Console Port Server SSH V_1.3.2 or later -> ssh2 will be used)
`ssh -t -1 blackbox:7001@scps` (openssh 3.1p1 or later - Secure Console Port Server V_1.3.2 or later -> ssh1 will be used)

To log in to a port that does not require authentication, the username is not necessary:

```
     ssh -t -2 :ttyS1@scps
```

Note: In this case, the file sshd_config must be changed in the following way:
PermitRootLogin Yes
PermitEmptyPassword Yes

Configuring sshd's client authentication using SSH Protocol version 1

1. Only RhostsAuthentication yes in sshd_config

- One of these:

  hostname or ipaddress in /etc/hosts.equiv or /etc/ssh/shosts.equiv

  hostname or ipaddress and username in ~/.rhosts or ~/.shosts and IgnoreRhosts no in sshd_config

- Client start-up command: ssh -t <System_ip or Serial_port_ip> (if the ssh client is running under a session belonging to a username present both in the workstation's database and the Secure Console Port Server SSH's database)

- Client start-up command: ssh -t -l <username> <System_ip or Serial_port_ip> (if the ssh client is running under a session belonging to a username present only in the workstation's database. In this case, the

<username> indicated would have to be a username present in the Secure Console Port Server SSH's database)

Note 1: Some ssh clients do not allow just this type of authentication, for security reasons.
Note 2: To access the serial port, the Secure Console Port Server SSH must be configured for local authentication.
Note 3: No root user should be used as username.

2. Only RhostsRSAAuthentication yes in sshd_config

- One of the RhostsAuthentication above settings

- Client machine's host key ($ETC/ssh_host_key.pub) copied into the Secure Console Port Server SSH /tmp/known_hosts file. The client hostname plus the information inside this file must be appended in one single line inside the file /etc/ssh/ssh_known_hosts or ~/.ssh/known_hosts and IgnoreUserKnownHosts no inside sshd_config. The following commands can be used for example:

```
echo –n “client_hostname “ >> /etc/ssh/ssh_known_hosts or ~/.ssh/known_hosts

cat /tmp/known_hosts >> /etc/ssh/ssh_known_hosts or ~/.ssh/known_hosts
```

- client start-up command: ssh -t <System_ip or Serial_port_ip>

Note 1: “client_hostname” should be the DNS name.
Note 2: To access the serial port, the Secure Console Port Server SSH must be configured for local authentication.
Note 3: No root user should be used as username.

3. Only RSAAuthentication yes in sshd_config

- Removal of Secure Console Port Server SSH's *.equiv, ~/.?hosts, and *known_hosts files

- client identity created by ssh-keygen and its public part (~/.ssh/identity.pub) copied into Secure Console Port Server SSH's ~/.ssh/authorized_keys

- client start-up command: ssh -t <System_ip or Serial_port_ip>

4. Only PasswdAuthentication yes in sshd_config

- Removal of Secure Console Port Server SSH's *.equiv, ~/.?hosts, *known_hosts, and *authorized_keys files

- client startup command: ssh –t -l <username> <System_ip or Serial_port_ip> or ssh –t –l <username:alias> <System_ip>

Configuring sshd's client authentication using SSH Protocol version 2

1. Only PasswdAuthentication yes in sshd_config DSA Authentication is the default (Make sure the parameter PubkeyAuthentication is enabled)

- Client DSA identity created by ssh-keygen -d and its public part (~/.ssh/id_dsa.pub) copied into Secure Console Port Server SSH's ~/.ssh/authorized_keys2 file

- Password Authentication is performed if DSA key is not known to the Secure Console Port Server SSH.
client start-up command: ssh -2 -t <System_ip or Serial_port_ip>

Notice:

All files "~/*" or "~/.ssh/*" must be owned by the user and readable only by others.

All files created or updated must have their full path and file name inside the file config_files and the command saveconf must be executed before rebooting the Secure Console Port Server SSH.

The Process Table

The process table shows which processes are running. Type ps -a to see a table similar to that below.

```
PID    Uid       State    Command
1      root      S        /sbin/inetd
31     root      S        /sbin/sshd
32     root      S        /sbin/cy_ras
36     root      S        /sbin/cy_wdt_led wdt led
154    root      R        /ps -a
```

To restart the cy_ras process use its process ID or execute the command:

```
signal_ras hup
```

This executes the ps command, searches for the cy_ras process id, then sends the signal HUP to the process, all in one step. Never kill cy_ras with the signals -9 or SIGKILL.

NTP Client Functionality

In order for the Secure Console Port Server SSH to work as a NTP (Network Timer Protocol) client, the IP address and either hostname or domain name of the NTP server must be set in the file /etc/hosts. The date and time will be updated from the NTP server after rebooting.

The Crond Utility

To use crond, first create the following two files for every process that it will execute:
1. crontab - the file that specifies frequency of execution, name of shell script, etc. should be set using the traditional crontab file format.

2. script shell - a script file with the Linux commands to be executed.

Next, create a line in the file /etc/crontab_files for each process to be run.
Each line must contain the three items:

- status (active or inactive) - if this item is not active, the script will not be executed.
- user - the process will be run with the privileges of this user, who must be a valid local user.
- source - pathname of the crontab file.

When the /etc/crontab_files file contains the following line:

```
active root /etc/tst_cron.src
```

and the /etc/tst_cron.src file contains the following line:

```
0-59 * * * * /etc/test_cron.sh
```

crond will execute the script listed in test_cron.sh with root privileges each minute.

Example files are in the /etc directory.

The next step is to update the system with the modified data in the files above and reboot the Secure Console Port Server SSH. Make sure the file named /etc/config_files contains the names of all files that should be saved to flash. Next, the command saveconf, which reads the /etc/config_files file, should then be run.

saveconf copies all the files listed in the file /etc/config_files from the ramdisk to /proc/flash/script. See step 5 in chapter 6 for more details.

The DHCP (Dynamic Host Configuration Protocol) Client

(Note: This feature is only available for firmware versions 1.2.x and above)
DHCP is a protocol that allows network administrators to assign IP addresses automatically to network devices. Without DHCP (or a similar protocol like BOOTP), each device would have to be manually configured. DHCP

automatically sends a new IP address to a connected device when it is moved to another location on the network. DHCP uses the concept of a fixed time period during which the assigned IP address is valid for the device it was assigned for. This “lease” time can vary for each device. A short lease time can be used when there are more devices than available IP numbers. For more information, see RFC 2131.

The DHCP client on the Ethernet Interface can be configured in two different ways, depending on the action the Secure Console Port Server SSH should take in case the DHCP server does not answer the IP address request:

1. No action is taken and no IP address is assigned to the Ethernet Interface (most common configuration):

- Set the global parameter **conf.dhcp_client** to **1**
- Comment all other parameters related to the Ethernet Interface (conf.eth_ip, etc.)
- Add the necessary options to the file /etc/network/dhcpcd_cmd (some options are described below)

2. The Secure COnsole Port Server SSH restores the last IP address previously provided in another boot and assigns this IP address to the Ethernet Interface:

- Set the global parameter **conf.dhcp_client** to **2**
- Comment all other parameters related to the Ethernet Interface (conf.eth_ip, etc.)
- Add the following lines to the file /etc/config_files:

```
/etc/network/dhcpcd_cmd
/etc/dhcpcd-eth0.save
```

- Add the option “-x” to the factory default content of the file /etc/network/dhcpcd_cmd:

```
/sbin/dhcpcd -x -c /sbin/handle_dhcp
```

- Add all other necessary options to the file /etc/network/dhcpcd_cmd (some options are described below)

In both cases if the IP address of the Secure Console Port Server SSH or the default gateway are changed, the Secure Console Port Server SSH will adjust the routing table accordingly.
Two files are related to DHCP:
/bin/handle_dhcp - the script which is run by the DHCP client each time an IP address negotiation takes place.

/etc/network/dhcpcd_cmd - contains a command that activates the DHCP client (used by the cy_ras program). Its factory contents are:

```
/sbin/dhcpcd -c /sbin/handle_dhcp
```

The options available that can be used on this command line are:

-D This option forces dhcpcd to set the domain name of the host to the domain name parameter sent by the DHCP server. The default option is to NOT set the domain name of the host to the domain name parameter sent by the DHCP server.

-H This option forces dhcpcd to set the host name of the host to the hostname parameter sent by the DHCP server. The default option is to NOT set the host name of the host to the hostname parameter sent by the DHCP server.

-R This option prevents dhcpcd from replacing the existing /etc/resolv.conf file.

The user should not modify the -c /sbin/handle_dhcp option.

Data Buffering

Since version 1.3.2 of the Secure Console Port Server SSH software, additional ramdisks can be created and used, for example, to buffer data. This removed the previous 700 kbyte restriction for all Secure Console Port Server SSH ports. Data buffering files are created in the directory /var/run/DB. Previously, data buffering files were named ttyS<nn>.data (where <nn> is the port number). Now, if the parameter s<nn>.serverfarm is configured for the port <nn>, this name will be used. For example, if the serverfarm is called bunny, the data buffering file will be named bunny.data.

The shell script /bin/build_DB_ramdisk creates a 4 Mbyte ramdisk for the Secure Console Port Server SSH 48-Port. Use this script as a model to create customized ramdisks for your environment. Any user-created scripts should be listed in the file /etc/user_scripts because rc.sysinit executes all shell scripts found there. This avoids changing rc.sysinit itself.

Data buffering can be done in local files or in remote files through NFS. When using remote files, the limitation is imposed by the remote Server (disk/partition space) and the data is kept in linear (sequential) files in the remote Server. When using local files, the limitation is imposed by the size of the available ramdisk.

The user may want to have data buffering done in file, syslog or both. For syslog, all.syslog_buffering and conf.DB_facility are the parameters to be dealt with as seen in the earlier chapters, and syslog-ng.conf file should be set accordingly (please see Appendix G for syslog-ng configuration file). For file, all.data_buffering is the parameter to be dealt with as seen in the early chapters.

Packet Filtering using ipchains

(Note: This feature is only available for firmware versions 1.2.x and above)
The Secure Console Port Server SSH uses the Linux utility ipchains to filter IP packets entering, leaving and passing through its interfaces. An ipchains tutorial is beyond the scope of this manual. For more information on ipchains, see the ipchains man page (not included with the Secure Console Port Server SSH) or the howto: http://netfilter.filewatcher.org/ipchains/HOWTO.html.

The syntax of the ipchains command is:

```
    ipchains -command chain [-s source] [-d destination] [-p protocol] [-j tar-
  get] [-i interface]
```

where **command** is one of the following:

A - Add a condition or rule to the end of the chain. Note that the order in which a condition appears in a chain can modify its application and the first rule added to a chain is processed first, etc.

D - Delete a condition from the chain. The condition must match exactly with the command's arguments to be deleted.

R- Replace a condition in the chain.

I - Insert a condition in a specified location in the chain.

L - List all conditions in the chain.

F - Flush (remove) all conditions in the chain.

N - Create a new chain.

X - Deletes a user-created chain

P - Policy applied for default handling

chain is one of the following:

input - filters incoming packets

output - filters outgoing packets

forward - filters packets which are not created by the Secure Console Port Server SSH and are not destined to the Secure Console Port Server SSH

user_created_chain - a previously defined (or in the process of being defined) chain created using the N command described above.

The output chain controls which packets are sent. A packet can be accepted by the input chain, but then rejected by the output chain. Likewise, the forward chain controls which packets will be routed. The input chain controls incoming packet filtering. The packet is either destined for the router or for another computer. In the latter case, the packet is processed by the forward chain. Packets that pass through the forward chain will then be processed by the output chain.

source and **destination** have the following format:

```
     [!]address[/mask] [!][port[:port]]
```

! : reverses the definition, resulting in the opposite.
address : host or network IP
port : defines a specific port
port:port : defines a range of ports
If a source or destination is not specified then 0.0.0.0/0 is used.

protocol is one of the following:
tcp, udp, icmp, all or a protocol number (see the file /etc/protocols for a list).

target is one of the following:
ACCEPT
DENY
the name of another chain

interface is:
eth0 (The Ethernet interface is the only option on the Secure Console Port Server SSH.) Lists do not need to be associated to an interface, so this option may be omitted.

To save changes made using the ipchains command, execute fwset. This command will save the filter configuration in the file /etc/network/firewall and then save the file in flash memory.

To delete the changes made (before fwset is executed) execute fwset restore to return to the lists previously saved in /etc/network/firewall. Only the lists previously saved using fwset will then be defined. This command is executed at boot to invoke the last configuration saved.
Another option is to edit the file /etc/network/firewall (or another file) directly, following the syntax defined in the file itself. If the file is edited in this way, the command fwset cannot be used to save and restore the configuration. Use
`ipchains-save > file_name` to save the lists in file_name

`updatefiles file_name` to save file_name to flash memory

`ipchains-restore < file_name` to restore the lists to the configuration in file_name

An example of the use of ipchains for a console access server

Referring to Fig 5.5

If the administrator wishes to restrict access to the consoles connected to the Secure Console Port Server SSH to a user on the workstation with IP address 200.200.200.4, a filter can be set up as shown below.

```
     ipchains -P input ACCEPT
     ipchains -P output ACCEPT
     ipchains -P forward ACCEPT
     ipchains -A input -p tcp -s ! 200.200.200.4 -d 0.0.0.0/0 23 -j DENY
     ipchains -A input -p tcp -s ! 200.200.200.4 -d 200.200.200.1 7001:7032 -j DENY
     ipchains -A input -p tcp -s ! 200.200.200.4 -d 0.0.0.0/0 22 -j DENY
```

ts_menu Script to Simplify telnet and ssh Connections

(Note: This feature is only available for firmware versions 1.2.x and above)

The ts_menu script can be used to avoid typing long telnet or ssh commands. It presents a short menu with the names of the servers connected to the serial ports of the Secure Console Port Server SSH. The server is selected by its corresponding number. ts_menu must be executed from a local session: via console, telnet, ssh, dumb terminal connected to a serial port, etc.

Only ports configured for console access (protocols socket_server or socket_ssh) will be presented.

To start having familiarity with this application, run ts_menu - h:

```
> ts_menu -h

USAGE: ts_menu options

-p        : Display Ethernet Ip and Tcp port
-i        : Display local Ip assigned to the serial port
-u <name> : Username to be used in ssh/telnet command
-U        : Allows choosing of different usernames for different
ports
-h        : print this help message
```

```
> ts_menu

Master and Slaves Console Server Connection Menu

1 65.186.161.113/TSJen800
2 65.186.161.82 /edson-r4.blackbox.com
3 65.186.161.84 /az84.blackbox.com
4 65.186.190.85
5 65.186.161.85 /az85.blackbox.com

Type 'q' to quit, a valid option [1-5], or anything else to refresh:
```

Selecting 1 in this example, and the user will access the local serial ports on that Secure Console Port Server SSH. In case the user selects 2 through 5, remote serial ports will be accessed. This is used when there is clustering (one Secure Console Port Server SSH master box and one ore more Secure Console Port Server SSH slave boxes).

In case the user selects 1 the possible screen to be displayed would be

```
Serial Console Server Connection Menu for your Master Terminal Server


1 ttyS1        2 ttyS2        3 s3serverfarm

Type 'q' to quit, 'b' to return to previous menu, a valid option[1-
3], or anything else to refresh:
```

Options 1 to 3 in this case are serial ports configured to work as CAS profile. Serial port 3 is presented as an alias name (s3serverfarm). When no name is configured in pslave.conf, ttyS<N> is used instead.

Once selected the serial port, the username and password for that port (in case there is a per user access to the port and -U is passed as parameter) will be presented. Otherwise, the acess is granted.

To access remote serial ports, the presentation will follow a similar approach as the one used for local serial ports.

The ts_menu script has the following line options:
-p : Displays Ethernet IP Address and TCP port instead of server names

```
Secure Console Port Server SSH: Serial Console Server Connection menu

1 209.81.55.79 7001 2 209.81.55.79 7002 3 209.81.55.79 7003
4 209.81.55.79 7004 5 209.81.55.79 7005 6 209.81.55.79 7006

Type 'q' to quit, a valid option [1-6], or anything else to refresh :
```

-i : Displays Local IP assigned to the serial port instead of server names

```
Secure Console Port Server SSH: Serial Console Server Connection menu

1 192.168.1.101 2 192.168.1.102 3 192.168.1.103 4 192.168.1.104
5 192.168.1.105 6 192.168.1.106

Type 'q' to quit, a valid option [1-6], or anything else to refresh :
```

-u <name> : Username to be used in ssh/telnet command. The default username is that used to log in to the Secure Console Port Server SSH.

-h : lists script options

APPENDIX B HARDWARE SPECIFICATIONS AND CABLING

General Hardware Specifications

The power requirements, environmental conditions and physical specifications of the Secure Cosole Port Server SSH are listed in the table below.

POWER SPECIFICATIONS Secure Console Port Server SSH						
	1-Port	4-Port	8-Port	16-Port	32-Port	48-Port
Input Voltage Range	External Universal Input Desktop Power Supply (100-240VAC auto-range input, 5VDC output)	External Universal Input Desktop Power Supply (100-240VAC auto-range input, 5VDC output)	External Universal Input Desktop Power Supply (100-240VAC auto-range input, 5VDC output)	Internal 100-240VAC auto-range (-48VDC option available)	Internal 100-240VAC auto-range (-48VDC option available)	Internal 100-240VAC auto-range
Input Frequency Range	50/60Hz	50/60Hz	50/60Hz	50/60Hz	50/60Hz	50/60Hz
Power @120VAC	5 W max	5 W max	6 W max	22 W max	26 W max	11 W max
Power @220VAC	6 W max	6 W max	8 W max	28 W max	37 W max	17 W max

ENVIRONMENTAL INFORMATION Secure Console Port Server SSH						
	1-Port	4-Port	8-Port	16-Port	32-Port	48-Port
Operating Temperature	40F to 104F (10°C to 40°C)	40F to 104F (10°C to 40°C)	40F to 104F (10°C to 40°C)	40F to 104F (10°C to 40°C)	40F to 104F (10°C to 40°C)	40F to 104F (10°C to 40°C)
Relative Humidity	10 to 90%, non-condensing	10 to 90%, non-condensing	10 to 90%, non-condensing	10 to 90%, non-condensing	10 to 90%, non-condensing	10 to 90%, non-condensing

PHYSICAL SPECIFICATIONS
Secure Console Port Server SSH

	1-Port	4-Port	8-Port	16-Port	32-Port	48-Port
External Dimensions	2.76in x 3.35 in x 1.18 in	8.5in x 4.75in x 1in	8.5in x 4.75in x 1in	17in x 8.5 in x 1.75 in	17in x 8.5 in x 1.75 in	17in x 8.5 in x 1.75 in
Weight	0.3 lb	1.5 lb	1.6 lb	6 lb	6.2 lb	8 lb

SAFETY
Secure Console Port Server SSH

	1-Port	4-Port	8-Port	16-Port	32-Port	48-Port
Approvals	FCC Class A, CE					

This section has all the information you need to quickly and successfully purchase or build cables to the Secure Console Port Server SSH. It focuses on information related to the RS-232 interface, which applies not only to the Secure Console Port server SSH but also to any RS-232 cabling. At the end of this chapter you will also find some information about the RS-485 interface, which is available in the Secure Console Port Server SSH 1-Port model only.

The RS-232 Standard

RS-232C, EIA RS-232, or simply RS-232 refer to a standard defined by the Electronic Industries Association in 1969 for serial communication. More than 30 years later, we have found more applications for this standard than its creators could have imagined. Almost all electronic devices nowadays have serial communication ports.

RS-232 was defined to connect Data Terminal Equipment, (DTE, usually a computer or terminal) to Data Communication Equipment (DCE, usually a modem):

DTE —> RS-232 —> DCE —> communication line –> DCE —> RS-232 –> DTE

RS-232 is now mostly being used to connect DTE devices directly (without modems or communication lines in between). While that was not the original intention, it is possible with some wiring tricks. The relevant signals (or wires) in a RS-232 cable, from the standpoint of the computer (DTE) , are:

Receive Data (RxD) and Transmit Data (TxD) – The actual data signals
Signal Ground (Gnd) - Electrical reference for both ends
Data Terminal Ready (DTR) - Indicates that the computer (DTE) is active
Data Set Ready (DSR) - Indicates that the modem (DCE) is active.
Data Carrier Ready (DCD) - Indicates that the connection over the communication line is active
CTS (Clear to Send, an input) – Flow control for data flowing from DTE to DCE
RTS (Request to Send, an output) – Flow control for data flowing from DCE to DTE

Not all signals are necessary for every application, so the RS-232 cable may not need all 7 wires.
The RS-232 interface defines communication parameters such as parity, number of bits per character, number of stop-bits and the baud rate. Both sides must be configured with the same parameters. That is the first thing to verify if you think you have the correct cable and things still do not work. The most common configuration is 8N1 (8 bits of data per character, no parity bit included with the data, 1 stop-bit to indicate the end of a character). The baud rate in a RS-232 line translates directly into the data speed in bits per second (bps). Usual transmission speeds range between 9,600 bps and 19,200bps (used in most automation and console applications) to 115,200 bps (used by the fastest modems).

Cable Length

The original RS-232 specifications were defined to work at a maximum speed of 19,200 bps over distances up to 15 meters (or about 50 feet). That was 30 years ago. Today, RS-232 interfaces can drive signals faster and through longer cables.
As a general rule, consider:

- If the speed is lower than 38.4 kbps, you are safe with any cable up to 30 meters (100 feet)
- If the speed is 38.4 kbps or higher, cables should be shorter than 10 meters (30 feet)
- If your application is outside the above limits (high speed, long distances), you will need better quality (low-

impedance, low-capacitance) cables.

Successful RS-232 data transmission depends on many variables that are specific to each environment. The general rules above are empirical and have a lot of safety margins built-in.

Connectors

The connector traditionally used with RS-232 is the 25-pin D-shaped connector (DB-25). Most analog modems and most older computers and serial equipment use this connector. The RS-232 interface on DB-25 connector always uses the same standard pin assignment.

The 9-pin D-shaped connector (DB-9) saves some space and is also used for RS-232. Most new PC COM ports and serial equipment (specially when compact size is important) uses this connector. RS-232 interfaces on DB-9 connectors always use the same standard pin assignment.

The telephone-type modular RJ-45 plug and jack are very compact, inexpensive and compatible with the phone and Ethernet wiring systems present in most buildings and data centers. Most networking equipment and new servers use RJ-45 connectors for serial communication. Unfortunately there is no standard RS-232 pin assignment for RJ-45 connectors. Every equipment vendor has its pin assignment.

Most connectors have two versions. The ones with pins are said to be “male” and the ones with holes are said to be “female”.

RS-232 Signal	**Name/Function (Input/Output)**	**DB-25 pins (Standard)**	**DB-9 pins (Standard)**	**RJ-45 pins (Black Box)**
Chassis	Safety Ground	1	Shell	Shell
TxD	Transmit Data (O)	2	3	3
RxD	Receive Data (I)	3	2	6
DTR	Data Terminal Ready (O)	20	4	2
DSR	Data Set Ready (I)	6	6	8
DCD	Data Carrier Detect (I)	8	1	7
RTS	Request To Send (O)	4	7	1
CTS	Clear To Send (I)	5	8	5
Gnd	Signal Ground	7	5	4

Straight-Through vs. Crossover Cables

The RS-232 interface was originally intended to connect a DTE (computer, printer and other serial devices) to a DCE (modem) using a straight-through cable (all signals on one side connecting to the corresponding signals on the other side one-to-one). By using some “cabling tricks”, we can use RS-232 to connect two DTEs as is the case in most modern applications.

A crossover (a.k.a. null-modem) cable is used to connect two DTEs directly, without modems or communication lines in between. The data signals between the two sides are transmitted and received and there are many variations on how the other control signals are wired. A “complete” crossover cable would connect TxD with RxD, DTR with DCD/DSR, and RTS with CTS on both sides. A “simplified” crossover cable would cross TxD and RxD and locally short-circuit DTR with DCD/DSR and RTS with CTS.

Which Cable Should be Used

First, look up the proper cable for your application in the table below. Next, purchase standard off-the-shelf cables from a computer store or cable vendor. For custom cables, refer to the cable diagrams to build your own cables or order them from Black Box or a cable vendor.

To Connect To	Use Cable
DCE DB-25 Female (standard) - Analog Modems - ISDN Terminal Adapters	Cable 1 – RJ-45 to DB-25 M straight-through (Custom) This custom cable can be ordered from Black Box or other cable vendors. A sample is included with the product ("straight-through").
DTE DB-25 Male or Female (standard) - Serial Terminals - Old PC COM ports - Most serial printers - Some Console Ports - Most automation devices	Cable 2 – RJ-45 to DB-25 F/M crossover (Custom) This custom cable can be ordered from Black Box or other cable vendors. A sample is included with the products ("Console").
DTE DB-9 Male or Female (standard) - Newer PC COM ports - Most Mice and pointing devices - Some automation devices	Cable 3 – RJ-45 to DB-9 F/M crossover (custom) This custom cable can be ordered from Black Box or other cable vendors. A sample is included with the products ("Console").
DTE RJ-45 Black Box (custom) - All Black Box Console Ports	Cable 4 – RJ-45 to RJ-45 crossover (custom) This custom cable can be ordered from Black Box or cable vendors using the provided wiring diagram.
DTE RJ-45 Netra (custom) - Sun Netra Console Ports - Cisco Console Ports	Cable 5- RJ-45 to RJ-45 crossover (custom) This custom cable can be ordered from Black Box or cable vendors using the provided wiring diagram.

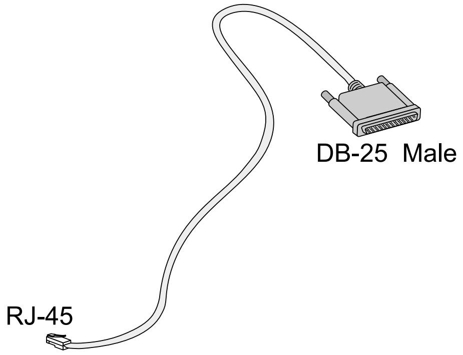
Cable Diagrams

Before using the following cable diagrams refer to the tables above to select the correct cable for your application. Sometimes, crossover cables are wired slightly differently depending on the application. A “complete” crossover cable would connect the TxD with RxD, DTR with DCD/DSR, and RTS with CTS across both sides. A “simplified” crossover cable would cross TxD and RxD and locally short-circuit DTR with DCD/DSR and RTS with CTS.

Most of the diagrams in this document show the “complete” version of the crossover cables, with support for modem control signals and hardware flow control. Applications that do not require such features have just to configure NO hardware flow control and NO DCD detection on their side. Both ends should have the same configuration for better use of the complete version of the cables.

Cable #1: RJ-45 to DB-25 Male, Straight Through

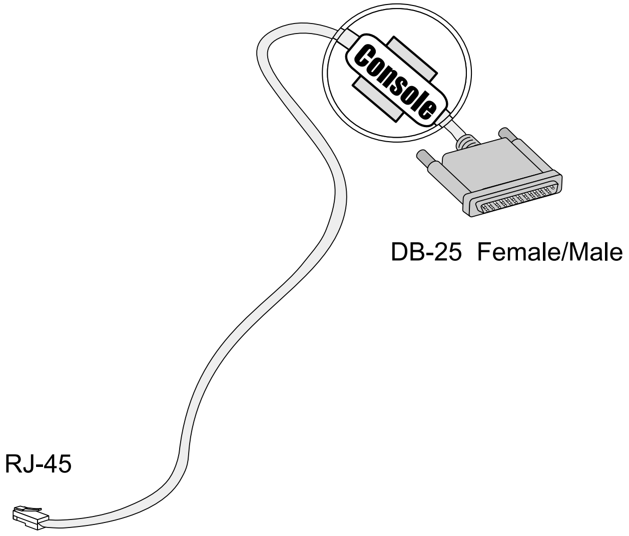
Application: It connects Secure Console Port Server SSH products (serial ports) to modems and other DCE RS-232 devices.

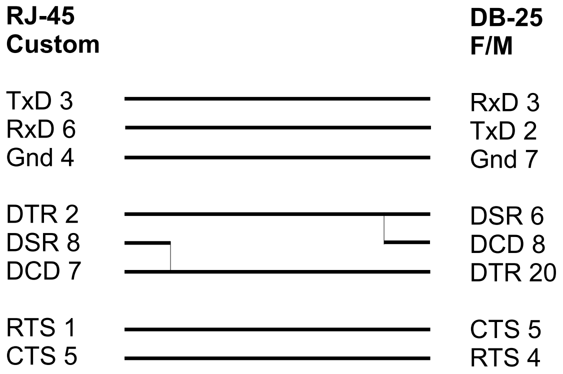


RJ-45 Male	**DB-25 Male**
TxD 3	TxD 2
RxD 6	RxD 3
Gnd 4	Gnd 7
DTR 2	DTR 20
DSR 8	DSR 6
DCD 7	DCD 8
RTS 1	RTS 4
CTS 5	CTS 5

Cable #2: RJ-45 to DB-25 Female/Male, Crossover

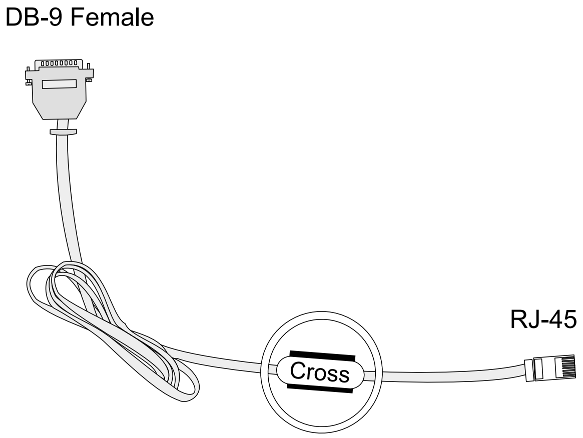
Application: It connects Secure Console Port Server SSH products (serial ports) to console ports, terminals, printers and other DTE RS-232 devices.

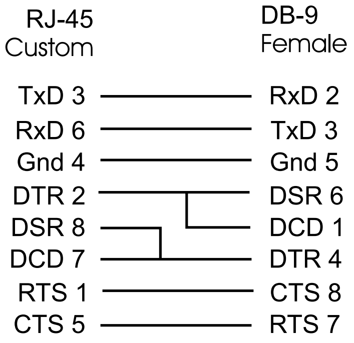




Cable #3: RJ-45 to DB-9 Female, Crossover

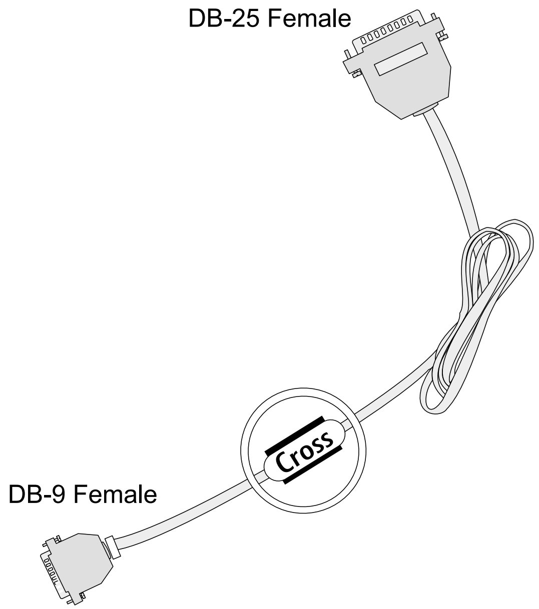
Application: It connects Secure Console Port Server SSH products (serial ports) to console ports, terminals, printers and other DTE RS-232 devices.

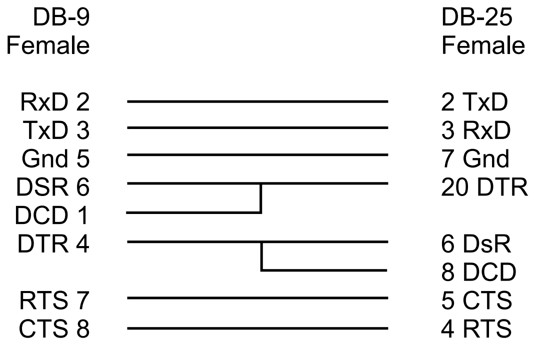




Cable #4: DB-9 Female to DB-25 Female, Crossover

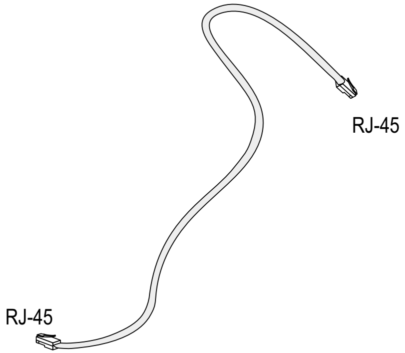
Application: It connects the Secure Console Port Server SSH 1-Port (serial port) to terminals, printers and other DTE RS-232 devices.

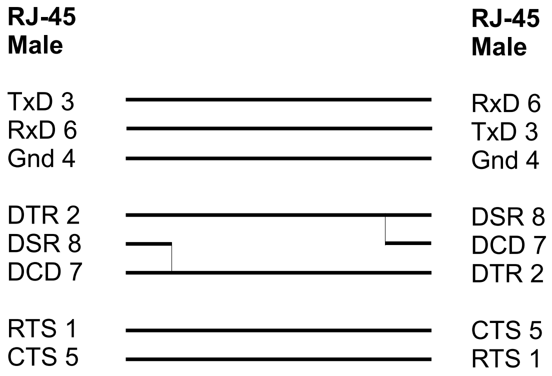




Cable #5: RJ-45 to RJ-45, Crossover

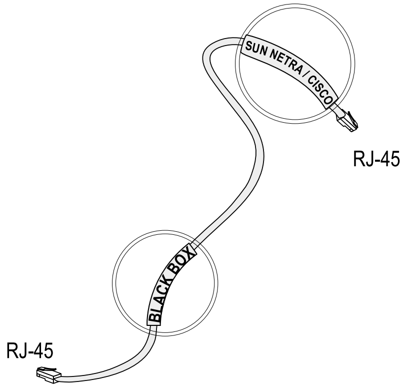
Application: Usually used to connect two ports of a Secure Console Port Server SSH product ("loopback") for testing purposes.





Cable #6: RJ-45 to Netra RJ-45, Crossover

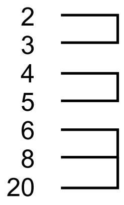
Usually used in console management applications to connect Secure Console Port Server SSH products to a Sun Netra server or to a Cisco product.

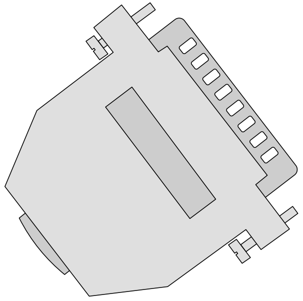


RJ-45 Custom	RJ-45 Netra
TxD 3	RxD 6
RxD 6	TxD 3
Gnd 4	Gnd 4
DTR 2	DSR 7
DCD 7	DTR 2
RTS 1	CTS 8
CTS 5	RTS 1

Loop-Back Connector for Hardware Test

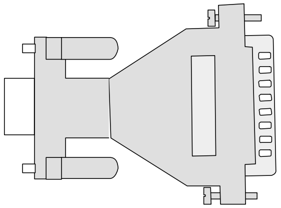
The use of the following DB-25 connector is explained in the Troubleshooting chapter.





DB-25 Male to DB-9 Female Adapter

The following adapter may be necessary.



DB-25	DB-9
2	3
3	2
4	7
5	8
6	6
7	5
8	1
20	4
22	9

Cabling Information Applicable only to the Secure Console Port Server SSH 1-Port

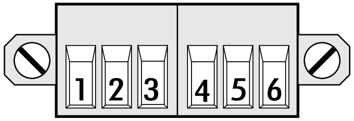
The RS-485 Standard

The RS-485 is another standard for serial communication and is available only in the Secure Console Port Server SSH 1-Port. Different from the RS-232, the RS-485 uses fewer wires - either two wires (one twisted pair) for half duplex communication or four wires (two twisted pairs) for full duplex communication. Another RS-485 characteristic is the “termination”. In a network that uses the RS-485 standard, the equipments are connected one to the other in a cascade arrangement. A “termination” is required from the last equipment to set the end of this network.

Secure Console Port Server Connectors

Although the RS-485 can be provided in different kinds of connectors, the Secure Console Port Server SSH 1-Port uses a 9-pin D-shaped connector (DB-9) and a block connector with the pin assignment described below.

RS-485 Signal	**Name/Function**	**DB-9 pins**	**Block connector pins**
Chassis	Safety Ground		1
TXD-	Transmit Data - (A)	7	2
TXD+	Transmit Data + (B)	3	3
RXD+	Receive Data + (B)	2	4
RXD-	Receive Data - (A)	8	5
Chassis	Safety Ground		6

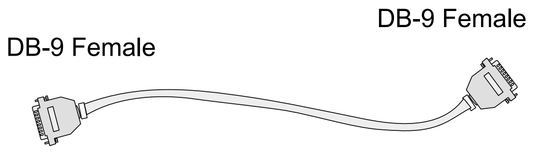


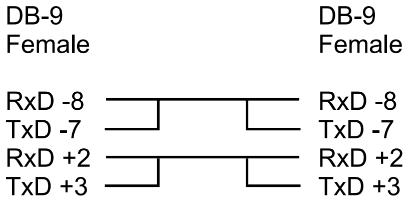
Notice that if the Secure Console Port Server SSH 1-Port is configured to use RS-485, the RS-485 signals will be available in both DB-9 and block connector. In this case, the DB-9 pins used in an RS-232 connection can be considered not connected.

Cable diagrams

Cable #1: DB-9 Female to DB-9 Female, Crossover half duplex

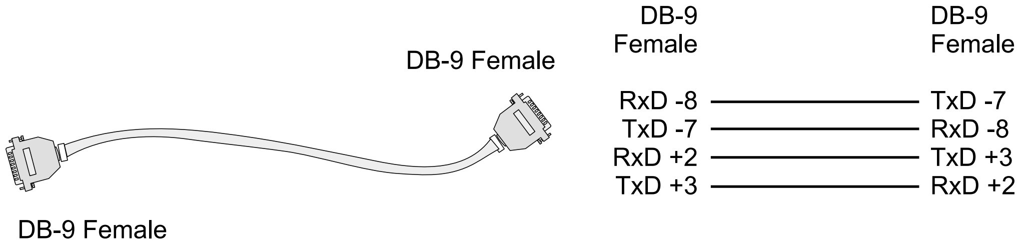
Application: It connects the Secure Console Port Server 1-Port (serial port) DTE RS-485 devices with half duplex communication.





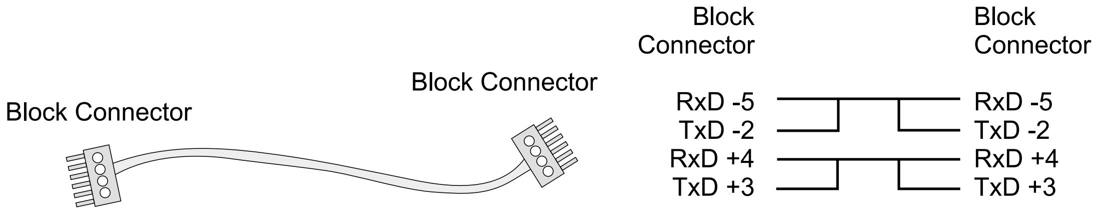
Cable #2: DB-9 Female to DB-9 Female, Crossover full duplex

Application: It connects the Secure Console Port Server SSH 1-Port (serial port) to DTE RS-485 devices with full duplex communication.



Cable #3: Block Connector to Block Connector, Crossover half duplex

Application: It connects the Secure Console Port Server SSH 1-Port (serial port) to DTE RS-485 devices with half duplex communication.



Cable #4: Block Connector to Block Connector, Crossover full duplex

Application: It connects the Secure Console Port Server SSH 1-Port (serial port) to DTE RS-485 devices with full duplex communication.

APPENDIX C SAMPLE PSLAVE.CONF FILES

The pslave.conf file with all possible parameters and their descriptions is presented first. The pslave.conf files for the three examples configured in chapter 6 follow.

The Complete pslave.conf File Provided with the Secure Console Port Server SSH

```
#
# pslave.conf   Sample server configuration file.
#
# The Terminal Server uses a virtual terminal concept.  Virtual terminals are
# named s1, s2, etc. Every virtual terminal should have a related
# physical device tty (without the "/dev/"). The tty parameter
# must be configured and must be unique for each virtual terminal.
#
# There two types of parameters:
#
# 1) Global parameters
#    These parameters have the prefix "conf." Example of global parameters
#    are ethernet ip address, etc.
#
# 2) Terminal Parameters.
#    These parameters have prefixes "all.", "s1.", "s2.", etc.
#
#    The "all." entries are used as a template for all virtual terminals.
#    Setting all.speed to 9600 will set all virtual terminal (s1, s2,
#    s3, etc.) speeds to 9600.
#
#    Note that you can change the "all." settings one by one.
#    If the parameter "s4.speed 19200" appears later in the file, all
terminals
#    except s4 will have speed 9600 bps and "s4" will have speed 19200 bps.
#

# Expansion Variables
#
# A list of format strings used by some parameters is provided here
# for reference.
#
# %l: login name
# %L: stripped login name
# %p: NAS port number
# %P: protocol
# %b: port speed
# %i: local IP
# %j: remote IP
# %1: first byte (MSB) of remote IP
# %2: second byte of remote IP
# %3: third byte of remote IP
# %4: fourth (LSB) byte of remote IP
# %c: connect-info
# %m: netmask
# %t: MTU
# %r: MRU
# %I: idle timeout
# %T: session timeout
# %h: hostname
# %%: %

# Generic SAMPLE:
# all async ports at 9600 bps, 8N1, no flow control
# Eth IP address 192.169.160.10/24 (MTU=1500)
# protocol socket_server
# host IP 192.168.160.8/24
# Radius Server IP 192.168.160.3 (authentication and accounting)
# authentication none
#

#
# Ethernet configuration.
#
# These parameters should only be configured in the file
# /etc/network/ifcfg_eth0  _IF_ the customer will not be using the
# cy_ras/portslave aplications. If the cy_ras/portslave aplications are _NOT_
# used put all ifconfig commands for the ethernet directly in the
# /etc/network/ifcfg_eth0.
#
# The cy_ras application OVERWRITES the ifcfg_eth0 file with the
# values configured here.
#
# The Secure COnsole Port Server SSH can request all of its ethernet
parameters to a DHCP server.
```

```
# The administrator can activate the dhcp client with more options changing
# the file /etc/network/dhcpcd_cmd.
#
# Valid values 0: DHCP disabled (default)
#              1: DHCP active
#              2: DHCP active and the Secure Console Port Server saves in
flash the last ip assigned
#                 by the DHCP server. This option requires changes in the
#                 files /etc/config_files and /etc/network/dhcpcd_cmd
#
#                 SEE Secure COnsole Port Server SSH manual for more
information.
#
#conf.dhcp_client       1

conf.eth_ip     192.168.160.10
conf.eth_mask    255.255.255.0
conf.eth_mtu1500

#
# Secondary IP address of ethernet
#

#conf.eth_ip_alias   192.168.161.10
#conf.eth_mask_alias 255.255.255.0

#
# Remote Network File System where data buffering will be written instead
# of the default directory '/var/run/DB'. The directory tree to which the
# file will be written must be NFS-mounted.
#
# If data buffering is turned on for port 1, for example, the data will be
# stored in /tmp/ts_data_buffer/{ttyS1.data | serverfarm} on the machines
# with IP address 192.168.160.11. The remote host must have NFS installed
# and the administrator must create, export and allow reading/writing to
# this directory.
# The size of this file is not limited by the value of the parameter
# s1.data_buffering, though the value cannot be zero since a zero value turns
# off data buffering.
#
#conf.nfs_data_buffering 192.168.160.11:/tmp/ts_data_buffer
```

```
#
# Lock directory - The lock directory is /var/lock for the Secure Console Port
Server SSH.
#   It should not be changed unless the user decides to customize the
#   operating system.
#
conf.lockdir/var/lock
#
# Location of the rlogin binary that accepts the "-i" flag.
#
conf.rlogin /usr/local/bin/rlogin-radius
#
# Location of our patched pppd with Radius linked in.
#
conf.pppd   /usr/local/sbin/pppd
#
# Location of the telnet utility. This can be the system telnet. (Optional)
#
conf.telnet /bin/telnet
#
# Location of ssh utility. This can be the system SSH. (Optional)
#
conf.ssh    /bin/ssh
#
# This parameter is only necessary when authentication is being
# performed for a port.  When set to one, it is possible to log
# in to the Terminal Server directly
# by placing a "!" before your login name, then using your normal
# password. This is useful if the Radius authentication server is down.
#
conf.locallogins 1
#

#
# Syslog facility for portslave
#
conf.facility 7

#
# Syslog facility for Data Buffering and Alarm
#
conf.DB_facility 7
```

```
#
# User groups make the configuration of Port access restrictions
# easier. The parameter s<nn>.users, that will be explained later,
# can be configured using a combination of group names and user names.
#
#conf.group mkt: paul, sam
#
#conf.group adm: joe, mark
#
#s1.users mkt, joe
#
#s2.users adm, sam

#
# Speed. All ports are set to 9600 baud rate, 8 bits, No parity, 1 stop bit.
# These values can be changed port by port later in the file.
#
all.speed       9600
all.datasize    8
all.stopbits    1
all.parity      none

#
# Media type - define media type and operation mode (half/full) duplex.
#
# valid values:
#   rs232                  - RS232 (default value).
#   rs485_half             - RS485 half duplex without terminator
#   rs485_full             - RS485 full duplex without terminator
#   rs485_half_terminator  - RS485 half duplex with terminator
#   rs485_full_terminator  - RS485 full duplex with terminator
#   rs422                  - alike rs485_full
#   rs422_terminator       - alike rs485_full_terminator

#all.media rs232


#
# Authentication type - either "local", "radius", "none", "remote"
# "local/radius", "radius/local", or "RadiusDownLocal".
#
```

```
# If the authentication type is configured as "local/radius" the portslave
# first tries to authenticate locally.  If it fails, portslave will try to
# authenticate using the radius server.
#
# If the authentication type is configured as "RadiusDownLocal" the portslave
# first tries to authenticate using the radius server.  If the Radius server
# sends back a rejection, authentication will fail.  Local authentication
# will be tried only if the Radius server is down (timeout).
#
all.authtype    none
#
# Authentication host and accounting host.  Two of each can be configured
# per port. The first is tried 'radretries' times before the
# second is tried. If 'radretries' is not configured, 5 is used by default.
# The parameter 'radtimeout' sets the timeout per query in seconds.
#
all.authhost1   192.168.160.3
all.accthost1   192.168.160.3
all.radtimeout  3
all.radretries  5
#all.authhost2  192.168.160.4
#all.accthost2  192.168.160.4
#
# The shared secret used by RADIUS.
#
all.secret  black box

#
# Default protocol.
#
# Valid values are
#  RAS profile: "slip", "cslip", "ppp", "ppp_only"
#  TS  profile: "login", "rlogin", "telnet", # "ssh", "ssh2", "socket_client"
#  CAS profile: "socket_server", "socket_ssh", "raw_data"
#
#  ppp_only ==> PPP over leased lines (only authentication PAP/CHAP)
#
#  ppp      ==> PPP with terminal post dialing (Auto detect PPP)
#

#
# Default ip address of linux host to which the terminals will connect.
# Used by the protocols rlogin, ssh, socket_client, etc.
#
all.host    192.168.160.8

#
# IP Address assigned to the serial port.
# The '+' after the value causes the interfaces to have
# consecutive ip addresses. Ex. 192.168.1.101, 192.168.1.107, etc.
#
# The IP number of a port is used when the RADIUS
# server does not send an IP number, or if it tells us to use a dynamic IP no.
#
all.ipno    192.168.1.101+
all.netmask 255.255.255.255

#
# Maximum reception/transmission unit size for the port
#
all.mtu     1500
all.mru     1500

#
# Standard message issued on connect.
#
all.issue       \r\n\
                TSLINUX - Portslave Internet Services\n\
\r\n\
     Welcome to terminal server %h port S%p \n\
\r\n\

#
# Login prompt.
#
all.prompt  %h login:

#
# Terminal type, for rlogin/telnet sessions.
#
all.term    vt100
```

```
#
# If you want the Terminal Server to update the
# login records (written to the /var/run/utmp and/or /var/log/wtmp
# files), set sysutmp/syswtmp to 1. This is useful for tracking
# who has accessed the Terminal Server and what they did.
#
all.sysutmp 1
all.syswtmp 0
all.utmpfrom    "%p:%P.%3.%4"

#
# Use initchat to initialize the modem.
#
# d == delay (1 sec), p == pause (0.1 sec), l == toggle DTR
# r == <CR>, l == <LF>
#
#all.initchat    TIMEOUT 10 \
#       "" \d\l\dATZ \
#       OK\r\n-ATZ-OK\r\n "" \
#       TIMEOUT 10 \
#       "" ATM0 \
#       OK\r\n "" \
#       TIMEOUT 3600 \
#       RING "" \
#       STATUS Incoming %p:I.HANDSHAKE \
#       "" ATA \
#       TIMEOUT 60 \
#       CONNECT@ "" \
#       STATUS Connected %p:I.HANDSHAKE

#
# Serial port flow control:
#    hard - hardware, rts/cts
#    soft - software, CTRL-S / CTRL-Q
#    none.
#
all.flow    none

#
#  DCD signal (sets the tty parameter CLOCAL). Valid values are 0 or 1.
#     In a socket session, if all.dcd=0, a connection request (telnet or
#     ssh) will be accepted regardless of the DCD signal and the connection
#     will not be closed if the DCD signal is set to DOWN.
#     In a socket connection, if all.dcd=1 a connection request will be
#     accepted only if the DCD signal is UP and the connection (telnet or
#     ssh) will be closed if the DCD signal is set to DOWN.
#
all.dcd     0

#
# PPP options - used if a PPP session is autodetected.
# Note that mru and mtu are both set to the MTU setting.
# Callback server is enabled when cb-script parameter is set.
#
#all.autoppp%i:%j novj \
#       proxyarp modem asyncmap 000A0000 \
#       noipx noccp login auth require-pap refuse-chap \
#       mtu %t mru %t \
#       ms-dns 192.168.160.5 ms-dns 0.0.0.0 \
#       cb-script /etc/portslave/cb_script \
#       plugin /usr/lib/libpsr.so

#
# PPP options - User already authenticated and service type is PPP.
#
#all.pppopt %i:%j novj \
#       proxyarp modem asyncmap 000A0000 \
#       noipx noccp mtu %t mru %t netmask %m \
#       idle %I maxconnect %T \
#       ms-dns 192.168.160.5 ms-dns 0.0.0.0 \
#       plugin /usr/lib/libpsr.so
#

#
# When not set to zero, this parameter sets the wait for a TCP connection
# keep-alive timer. If no traffic passes through the Terminal Server for
# this period of time (ms), the Terminal Server will send a modem statuss
# message to the remote device to see if the connection is still up.
#
#all.poll_interval   1000

#
# Transmission interval - Controls the interval between two consecutive datas
#                         packets transmited to the Ethernet. Only valid for
#                         protocols socket_server, raw_data, and
socket_client.
#
# Valid values : 0 - transmit packet immediately (no interval).
#               10, 20, 30, ... interval in milliseconds.
#
#all.tx_interval 100

#
# Inactivity timeout - Defines the time in minutes that a conection can
#                      remains without activity (rx/tx). Only for CAS profile
#                      and socket_client protocol.
#
#all.idletimeout 5

# This defines an alternative labeling system for the Terminal Server ports.
# This parameter is used by the protocols telnet, socket_client and
# socket_server. It is mandadory if the protocol is socket_server, otherwise
# 23 will be used.
#
# The '+' after the numerical value causes the interfaces to be numbered
# consecutively.  Ex. 7001, 7002, 7003, etc.
#
all.socket_port      7001+

# Data buffering configuration
#
# A non-zero value activates data buffering. The number is equal to the
# buffer size. A file /var/run/DB/{ttyS#.data | serverfarm} is created on
# the Secure Console Port Server SSH and all data received from the port is
captured.
# The files for all buffered ports combined can contain up to the amount
# of available memory in the ram disk. This amount can be discovered
# by typing: "df<enter>".
# Each file is a revolving file which is overwritten as the limit of buffer
# size is reached. These files can be viewed using the normal Unix tools
# (cat, vi, more, etc.).
# If there is not enough available ram disk, NFS_buffering can be used.  There
# is effectively no limit to NFS buffer size.
#
all.data_buffering 0
```

```
#
# When non-zero, the contents of the data buffer are sent to the syslog
# server every time a quantity of data equal to this parameter is collected.
# [40 to 255 recomended]
#
all.syslog_buffering 0

# Alarm configuration
# When non zero, all data received from the port is captured and is sent to
syslog-ng
# with LOCAL [0+DB_facility] facility and INFO level.
# The syslog-ng.conf file should be set accordingly to make an action
# (please see the documentation).
#
all.alarm 0

#
# Controls the presentation of the Data buffering menu
#
# MENU:
# "A non-empty Data Buffering File was found. Choose wich action
#  should be performed ( (I)gnore, (D)isplay, (E)rase or (S)how and erase ) :"
#
#  valid values:
#   0 - Shows the menu with all options.
#   1 - Doesn't show the menu and any non empty data buffering file
#   2 - Doesn't show the menu but shows a non empty data buffering file
#   3 - Shows the menu without the options "erase" and "show and erase".
#
#all.dont_show_DBmenu 1

#
# Send Break to the TTY when this string is received (ssh only).
#
all.break_sequence ~break

#
# Authentication of Radius users registered without passwords
#
# When enabled (value 1) and a user registered in
# the Radius database with a blank password tries to log in, the user
# is authenticated.  This is a very weak level of security since
# a user would only need to know that a particular username exists.
# This does not affect Radius users registered with passwords.
#
all.radnullpass 0

#
# Automatic User Definition (more useful when used to a specific port)
#
# This parameter is only used if the port is configured as a Terminal Server
# (login, telnet, rlogin, ssh and ssh2) and authentication type 'none'.
#
#all.userauto edson

#
# Port access restriction (more useful when used to a specific port).
#   A single comma and spaces/tabs may be used between names.
#   A comma may not appear between the ! and the first user name.
#   The users may be local or Radius.
#
# In this example, the users joe and mark CANNOT access any serial port
#
#all.users ! joe, mark
#
# In this example, ONLY the users joe and mark CAN access any serial port
#
#all.users   joe, mark

#
# Serverfarm is an alias name for a server connected to the Secure Console
Port Server SSH
# through one of its serial ports (only useful if assigned to a specific
port).
# This alias is used as name to the data buffering file and in ssh command to
# select a serial port that should be configured as "socket_ssh".
#
# The value entered here should be the same used in the ssh command. Ex.
#
# ssh -t <username>:<server_connected_to_serial1>@<tsname>  or
# ssh -t -l <username>:<server_connected_to_serial1> <tsname>
#
#s1.serverfarm server_connected_to_serial1
```

```
#
# Snif session mode (in, out, i/o). With this parameter the user can select
# which data will be sent to the monitor. The default is "out".
#
all.sniff_mode out

#
# Users that are allowed to sniff sessionsI (administrator). This field has
# the same format "all.users", but the '!' should be used used with
PRECAUTION.
#
# In this example, ONLY the users joe, mark, and peter CAN access any
# serial port (to create first session) but ONLY the user peter can
# sniff or cancel another session.
#
#all.users           joe, mark
#all.admin_users peter

#
# Port-specific parameters
#
s1.tty      ttyS1
s2.tty      ttyS2
s3.tty      ttyS3
s4.tty      ttyS4
s5.tty      ttyS5
s6.tty      ttyS6
s7.tty      ttyS7
s8.tty      ttyS8
s9.tty      ttyS9
s10.tty     ttyS10
s11.tty     ttyS11
s12.tty     ttyS12
s13.tty     ttyS13
s14.tty     ttyS14
s15.tty     ttyS15
s16.tty     ttyS16

# for Secure Console Port Server SSH 32-Port uncomment s17 through s32
#s17.tty    ttyS17
#s18.tty    ttyS18
#s19.tty    ttyS19
```

```
#s20.tty    ttyS20
#s21.tty    ttyS21
#s22.tty    ttyS22
#s23.tty    ttyS23
#s24.tty    ttyS24
#s25.tty    ttyS25
#s26.tty    ttyS26
#s27.tty    ttyS27
#s28.tty    ttyS28
#s29.tty    ttyS29
#s30.tty    ttyS30
#s31.tty    ttyS31
#s32.tty    ttyS32

# for Secure Console Port Server SSH uncomment s33 through s48
#s33.tty    ttyS33
#s34.tty    ttyS34
#s35.tty    ttyS35
#s36.tty    ttyS36
#s37.tty    ttyS37
#s38.tty    ttyS38
#s39.tty    ttyS39
#s40.tty    ttyS40
#s41.tty    ttyS41
#s42.tty    ttyS42
#s43.tty    ttyS43
#s44.tty    ttyS44
#s45.tty    ttyS45
#s46.tty    ttyS46
#s47.tty    ttyS47
#s48.tty    ttyS48
```

The pslave.cas File Provided With the Secure Console Port Server SSH for the Console Access Server Example

```
#
# pslave.conf   Sample server configuration file.
#
# Console Access Server Profile
#


conf.eth_ip     200.200.200.1
conf.eth_mask    255.255.255.0
conf.eth_mtu1500
#conf.nfs_data_buffering 192.168.160.11:/tmp/ts_data_buffer
conf.lockdir/var/lock
conf.facility 7

all.speed       9600
all.datasize    8
all.stopbits    1
all.parity      none
all.authtype    radius
all.authhost1   200.200.200.2
all.accthost1   200.200.200.2
all.radtimeout  3
all.radretries 5
all.secret      black box
all.ipno    192.168.1.101+
all.term    vt100
all.issue       \r\n\
                TSLINUX - Portslave Internet Services\n\
\r\n\
     Welcome to terminal server %h port S%p \n\
\r\n\

all.prompt  %h login:
all.term        vt100
all.flow    hard
all.poll_interval    0
all.socket_port      7001+
all.protocol socket_server
all.data_buffering 0
all.syslog_buffering 0
#all.dont_show_DBmenu 1

#
# Users joe and mark will only have access granted to the serial port ttyS2
#
all.users ! joe, mark

#
# Sniff sessions will only display data sent by servers connected
# to the serial port.
#
all.sniff_mode out

#
# Only users peter and john can open a sniff session
#
all.admin_users peter, john

#
# Port-specific parameters
#

#-----------------
# PORT 1
#-----------------

s1.tty        ttyS1
s1.authtype   local
s1.serverfarm server_connected_serial1

#-----------------
# PORT 2
#-----------------
s2.tty      ttyS2
s2.users   joe, mark
s2.protocol   socket_ssh
```

```
#-----------------
# PORT 8
#-----------------

s8.tty          ttyS8
s8.protocol     socket_ssh
s8.authtype     none
s8.serverfarm   server_connected_serial8
```

The pslave.ts File provided with the Secure Console Port Server SSH for the Terminal Server

Example

```
#
# pslave.conf   Sample server configuration file.
#
#  Terminal Server Profile


conf.eth_ip     200.200.200.1
conf.eth_mask    255.255.255.0
conf.eth_mtu1500
conf.lockdir/var/lock
conf.rlogin /usr/local/bin/rlogin-radius
conf.telnet /bin/telnet
conf.ssh    /bin/ssh
conf.locallogins 0

all.speed       9600
all.datasize    8
all.stopbits     1
all.parity      none
all.authtype    none
all.protocoltelnet
all.host    200.200.200.3
all.issue       \r\n\
                TSLINUX - Portslave Internet Services\n\
\r\n\
     Welcome to terminal server %h port S%p \n\
\r\n\


all.prompt  %h login:
all.term    vt100
all.flow    hard
all.socket_port 23

#
```

```
# Users joe and mark will only have access to serial port ttyS5
#
all.users ! joe, mark

#
# Port-specific parameters
#
s1.tty          ttyS1

s2.tty          ttyS2
s2.authtype     local
s2.protocol     rlogin
s2.speed        19200
s2.datasize     7
s2.stopbits     2
s2.parity       even

s3.tty          ttyS3
s3.protocol     ssh2
s3.authtype     remote

s4.tty          ttyS4
s4.protocol     ssh
s4.authtype     remote

s5.tty          ttyS5
s5.users   joe, mark
```

The pslave.ras File Provided With the Secure Console Port Server SSH for the Remote Access Server Example

```
#
# pslave.conf    Sample server configuration file.
#
# Remote Access Server Profile
#

conf.eth_ip 200.200.200.1
conf.eth_mask    255.255.255.0
conf.eth_mtu1500
conf.lockdir/var/lock
conf.pppd   /usr/local/sbin/pppd-radius
conf.facility 7
all.speed       57600
all.datasize    8
all.stopbits    1
all.parity      none
all.authtype    radius
all.authhost1    200.200.200.2
all.accthost1    200.200.200.2
all.radtimeout   5
all.radretries  5
all.secret  cocomero
all.protocolppp
all.ipno    200.200.200.11+
all.netmask 255.255.255.255
all.mtu     1500
all.mru     1500
all.issue       \r\n\
               TSLINUX - Portslave Internet Services\n\
\r\n\
     Welcome to terminal server %h port S%p \n\
\r\n\
all.initchat    TIMEOUT 10 \
        "" \d\l\dATZ \
        OK\r\n-ATZ-OK\r\n "" \
        "" ATMO \
        OK\R\N "" \
        TIMEOUT 3600 \
        RING "" \
        STATUS Incoming %p:I.HANDSHAKE \
        "" ATA \
        TIMEOUT 60 \
        CONNECT@ "" \
        STATUS Connected %p:I.HANDSHAKE


all.flow    hard
all.dcd     1
all.autoppp %i:%j novj \
        proxyarp modem asyncmap 000A0000 \
        noipx noccp login auth require-pap refuse-chap \
        mtu %t mru %t \
        plugin /usr/lib/libpsr.so

all.pppopt  %i:%j novj \
        proxyarp modem asyncmap 000A0000 \
        noipx noccp mtu %t mru %t netmask %m \
        idle %I maxconnect %T \
        plugin /usr/lib/libpsr.so

#
# Port-specific parameters
#

#------------------------------------------------
# PORT 1 PPP dial in with terminal post dialing
#------------------------------------------------
s1.tty      ttyS1

#------------------------------------------------
# PORT 2 PPP dial in with terminal post dialing
#------------------------------------------------
s2.tty      ttyS2
s2.authtype local/radius
```

```
#-------------------------------------------
# PORT 3 PPP Leased line
#-------------------------------------------
s3.tty      ttyS3
s3.protocol ppp_only
s3.pppopt   %i:%j novj \
        proxyarp modem asyncmap 000A0000 \
        noipx noccp login auth require-pap refuse-chap \
        mtu %t mru %t \
        plugin /usr/lib/libpsr.so
s3.initchat ""
s3.issue    ""
```

APPENDIX D CUSTOMIZATION

Everything related to the Secure Console Port Server SSH can be traced back to two files: `/etc/rc.sysinit` and `/etc/inittab`. All Secure Console Port Server SSH application programs are started during boot by the init process. The related lines in the `/etc/inittab` file are listed below:

```
# System initialization.
::sysinit:/etc/rc.sysinit

# Single user shell
#console::respawn:/bin/sh < /dev/console > /dev/console 2> /dev/console
ttyS0::respawn:/sbin/getty -p ttyS0 ansi
::respawn:/sbin/cy_wdt_led wdt led

# Black Box RAS
::once:/sbin/cron
::once:/sbin/snmpd
::once:/sbin/cy_buffering
::once:/sbin/cy_ras
::once:/sbin/sshd -f /etc/ssh/sshd_config
::once:/sbin/ex_ntpclient
::once:/bin/webs
::once:/bin/syslog-ng
::once:/bin/cy_alarm
::wait:/sbin/fwset restore
```

To customize the Secure Console Port Server SSH, change these lines or add others.

For instance, to disable WEB services, comment the line referring to webs as follows.

#::once:/bin/webs

If the /etc/inittab file is changed, edit the /etc/config_files file and add a line containing only “/etc/inittab”. Save the file and exit the editor. Save the new configuration by executing saveconf. Then, the Secure Console Port Server SSH should be turned off and then turned on again. This is necessary because the init program provided by Busybox, a tool that emulates rm, cp, etc., but uses much less space, does not support the option ‘q’.

Black Box provides a development kit which allows changes to be made to the Secure Console Port Server SSH’s software. However, Black Box does not provide free technical support for systems modified in this way. Any changes are the responsibility of the user.

APPENDIX E MULTIPLE SNIFFING

Versions 1.3.2 and earlier

Secure Console Port Server SSH allows a maximum of 2 connections to each serial port, as follows:

- 1 common session: user can execute read and write commands to the tty port. Session can be established by a regular user or by an administrator.

- 1 sniffer session: user can execute only read commands, in order to monitor what is going on in the other (main) session. Session can only be established by an administrator, defined by the parameter all.admin_users or sN.admin_users in the file pslave.conf (exception: authentication none - anyone can open a sniffer).

The first connection always opens a common session. After the second connection has been established and the user is authenticated, the Secure Console Port Server SSH shows the following menu to the administrator user:

```
*
* * * ttySN is being used by (<user_name>) !!!
*
1 - Assume the main session
2 - Initiate a sniff session
3 - Quit
Enter your option :
```

If the second user is not an administrator, his connection is automatically refused.

This description is valid for all of the available protocols (socket_server, socket_ssh or raw_data).

Versions 1.3.3 and later

Users will be able to open more than one common and sniff sessions at the same port. For this purpose, the following configuration items will be included in the file pslave.conf:

- all.multiple_sessions: valid for all the serial ports; must be “yes” or “no”. The default value is “no”.
- sN.multiple_sessions: valid only for port N; must be “yes” or “no”. If it is not defined, it will assume the value of all.multiple_sessions.
- all.escape_char: valid for all the serial ports; this parameter will be used to present the menus below to the user. Only characters from ‘^a’ to ‘^z’ (i.e. CTRL-A to CTRL-Z) will be accepted. If this parameter is not set in pslave.conf, or in case it contains an invalid value, regular sessions will not be allowed to return to the menu regardless to what is typed by the user, whereas sniffer sessions will present the menu only if users type <CTRL-Z>. In addition, regular sessions will only be allowed to see the menu if the protocol used is “socket_server” or “socket_ssh”.
- sN.escape_char: valid only for port N; this parameter will be used to present the menus below to the user. Only characters from ‘^a’ to ‘^z’ (i.e. CTRL-A to CTRL-Z) will be accepted. If it is not defined, it will assume the value of all.escape_char.

When no multiple sessions are allowed for one port, the behavior of the Secure Console Port Server SSH when someone connects to it will be as described for version 1.3.2 and earlier. Otherwise, it will be as follows:

a. The first user to connect to the port will open a common session.

b. From the second connection on, only admin users will be allowed to connect to that port. The Secure Console Port Server SSH will send the following menu to these administrators (defined by the parameter all.admin_users or sN.admin_users in the file pslave.conf):

```
*
* * * ttySN is being used by (<first user name>) !!!
*
1 - Initiate a regular session
2 - Initiate a sniff session
3 - Send messages to another user
4 - Kill session(s)
5 - Quit
Enter your option :
```

If the user selects 1 - Initiate a regular session, he will share that serial port with the users that were previously connected. He will read everything that is received by the serial port, and will also be able to write to it.

If the user selects 2 - Initiate a sniff session, he will start reading everything that is sent and/or received by the serial port, according to the parameter all.sniff_mode or sN.sniff_mode (that can be **in, out or i/o**).

When the user selects 3 - Send messages to another user, the Secure Console Port Server SSH will send the user’s messages to all the sessions, but not to the tty port. Everyone connected to that port will see all the “conversation” that’s going on, as if they were physically in front of the console in the same room. These messages will be formatted as [Message from user/PID] <<message text goes here>> by the Secure Console Port Server SSH.

To inform the Secure Console Port Server SSH that the message is to be sent to the serial port or not, the user will have to use the menu.

If the administrator chooses the option 4 - Kill session(s), the Secure Console Port Server SSH will show him a list of the pairs PID/user_name, and he will be able to select one session typing its PID, or “all” to kill all the sessions.

Option 5 - Quit will close the current session and the TCP connection.

Only for the administrator users: typing all.escape_char or sN.escape_char from the normal or sniff session or "send message mode" will make the Secure Console Port Server show the previous menu. If this parameter is not set in pslave.conf, or it contains an invalid value, the regular sessions will not be allowed to return to the menu, and the sniffer sessions will be able to do it typing <CTRL-Z>. In addition, the regular session will only be allowed to see the menu if the protocol used is "socket_server" or "socket_ssh".

APPENDIX F CONFIGURATION WIZARD

Using Wizard through CLI

The user has a choice to configure the Secure Console Port Server SSH using the standard vi editor. For those not familiar with the editor, there's a way to pre-configure the unit (just basic configuration such as IP address of the Secure Console Port Server SSH) using the CLI. After that, they can continue configuring the unit through the WEB.

Once using the WEB, there's a wizard button to set basic parameters for a given profile (CAS, TS or RAS) to speed up the configuration process. This is used by customers who want basic features for a given profile. Customers who want to explore the features for that profile (data buffering, session sniffing, etc) would use the WEB thoroughly.

The configuration wizard application is a quicker and easier way to configure the Secure Console Port Server SSH. The use of this application is recommended if you are not familiar with the vi editor or if you just want to do a quick configuration of the system.

The command 'wiz' gets you started with some basic configuration. After executing this, you can then use the web configuration manager to continue configurations for the system. The files that will be eventually modified if you decide to save to flash at the end of this application are:

1. /etc/hostname
2. /etc/hosts
3. /etc/resolv.conf
4. /etc/network/st_routes
5. /etc/portslave/pslave.conf

Type 'wiz' in your Secure Console Port Server SSH terminal.

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************

Ok, let's get started! I need a few basic information on the system
so that it can know where it is located within the network and it
can know about its neighbor or its local environment.


Set to defaults? (y/n) [N] :
```

FIGURE F.1

The default answer or value to any question is in the brackets. For figure F.1, either just hit ENTER to execute whatever is in between the brackets or type ‘n’ to NOT reset the current configurations to the Black Box defaults or type ‘y’ to reset to Black Box default configurations.

The configuration begins in the next screens. There are instructions on how to use the wizard on each screen. There is also an explanation of each parameter before asking for it. To use the rest of the ‘wiz’ application, follow the instructions:

The default or the current value for the parameter is displayed inside the brackets. Just hit ENTER if you are satisfied with the value in the brackets. If not, enter the appropriate parameter and press ENTER.

If at any time, you want to exit the wizard or you want to skip the rest of the configuration, press ESC. This will immediately display a summary of the current configuration for your verification before exiting the application.

```
*********************************************************************
*************  C O N F I G U R A T I O N   W I Z A R D  *************
*********************************************************************



INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.


HOSTNAME - An alias for your system.
This way you can always refer to the system by this name
rather than its IP address.

Hostname[Secure Console Port Server SSH]:
```

FIGURE F.2

```
********************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
********************************************************************



INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.



IP - The IP address of your system (on its Ethernet
interface.) This is the address of the system within your
network. See you network administrator to obtain a valid
IP address for the system.

IP of your system[192.168.160.10]:
```

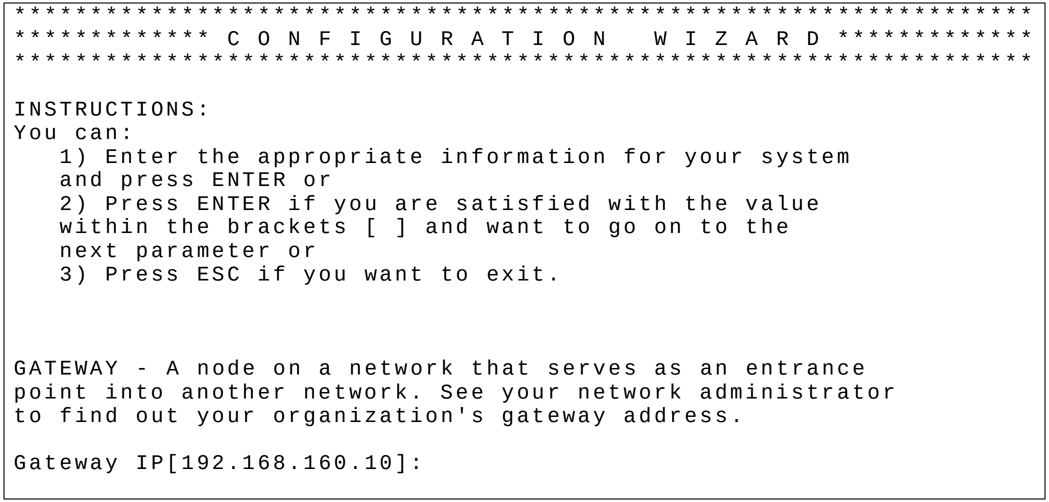
FIGURE F.3

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************


INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.


DOMAIN NAME - A name that locates or identifies your
organization within the Internet.

Domain name[mycompany.com]:
```

FIGURE F.4

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************



INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.


DOMAIN NAME SERVER - The IP address of the server that
resolves domain names. Your domain name is alphabetic so
that it is easier to remember. Everytime you see the domain
name, it is actually being translated into an IP address by
the domain name server. See your network administrator to
obtain this IP address for the domain name server.

Domain Name Server[127.0.0.1]:
```

FIGURE F.5



```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************


INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.




GATEWAY - A node on a network that serves as an entrance
point into another network. See your network administrator
to find out your organization's gateway address.

Gateway IP[192.168.160.10]:
```

FIGURE F.6

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************

INSTRUCTIONS:
You can:
   1) Enter the appropriate information for your system
   and press ENTER or
   2) Press ENTER if you are satisfied with the value
   within the brackets [ ] and want to go on to the
   next parameter or
   3) Press ESC if you want to exit.



NETMASK - A string of 0's and 1's that mask or screen out
the host part of an IP address so that only the network
part of the address remains.

Netmask[255.255.255.0]:
```

FIGURE F.7

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************



Your current configuration parameters are:

Hostname: Secure Console Port Server SSH
System IP: 192.168.160.10
Domain Name: mycompany.com
DNS: 127.0.0.1
Gateway: 192.168.160.10
Mask: 255.255.255.0

Are all these parameters correct (Y)es or (N)o [N] :
```

FIGURE F.8

Type ‘y’ if all parameters are correct. Type ‘n’ or just press ENTER if not all the parameters are correct and you want to go back and redo them.

If ‘n’ is entered, this is displayed:

```
Type 'c' to go back and CORRECT the current configuration
parameters or 'q' to QUIT:
```

Type ‘c’ to go back and CORRECT the current configuration parameters or ‘q’ to QUIT.

If ‘y’ is entered, Figure F.9 is displayed. This figure explains what saving to flash means. Type ‘y’ if you want to save to flash. Type ‘n’ if you don’t want to save to flash. You can now continue Secure Console Port Server SSH configuration using the web browser by typing in the IP address of the system. If you choose to not save to flash, all the new configuration will be lost if you were to reboot the system. However, all configuration will be kept if you saved to flash.

```
*******************************************************************
************* C O N F I G U R A T I O N   W I Z A R D *************
*******************************************************************



You can now use the browser to finish your system configu-
rations, but before that, please read below.
Flash refers to a type of memory that can be erased and
reprogrammed in units of memory known as blocks rather than
one byte at a time; thus, making updating to memory easier. If you
choose to save to flash, your configurations thus
far will still be in the memory of the system even after you
reboot it. If you don't save to flash and if you were to
reboot the system, all your new configurations will be lost
and you will have to reconfigure the system. Do you want to save
your configurations to flash (Y/N) [N]:
```

FIGURE F.9

NOTE: Using telnet to configure the Secure Console Port Server SSH, if you reconfigure the IP address of the Ethernet interface you are supposed to have your telnet connection lost. In that case, close the telnet client and reopen it using this time the new IP address configurated for that Secure Console Port Server SSH box.

Using Wizard through WEB

The web interface supports wizards for the serial ports configuration. The following profiles are supported for the serial ports:

- Console Access Server (CAS) profile
- Terminal Server (TS) profile
- Remote Access Server (RAS) profile
- Automation Profile (a subset of CAS profile, only for the Secuere Console Port Server SSH 1-Port)

Most of the applications should fit in one of these profiles, so the wizard is a useful tool to ease the configuration of the serial ports. The web interface will access the wizard files for each profile:

- /etc/portslave/pslave.wiz.cas (CAS profile)
- /etc/portslave/pslave.wiz.ts (TS profile)
- /etc/portslave/pslave.wiz.ras (RAS profile)
- /etc/portslave/pslave.wiz.auto (Automation profile)

The wizard configuration is set by pressing one of the buttons which appear in the Wizard section in the Serial Port Configuration page. The default parameters will then be set in the page and, after that, the user must change the parameters which need to be changed, and then press the Submit button.

The tables below show the values of the parameters which will be set for each profile. The parameters in bold are the ones whose value will probably be required to change.

Console Access Server (CAS) profile:

Parameter	Value	Comments
speed	**9600**	**Port speed - 9600bps**
datasize	8	8 bits
stopbits	1	1 stop bit
parity	none	no parity
flow	hard	hardware flow control
dcd	0	not sensitive to DCD signal
sysutmp	1	write the users in utmp log file
syswtmp	0	Do not write the users in wtmp log file.
authtype	radius	Authentication can be through Radius, TACACS+ or Local.
authhost1	**200.200.200.2**	**change it to the authentication server of your environment**
accthost1	**200.200.200.2**	**change it to the accounting server of your environment**
radtimeout	3	3 minutes timeou
secret	**black box**	**change it to the secret of the RADIUS/ TACACS+ server**
radretries	5	5 retries before giving up authenticating
radnullpass	0	Don't allow users with null passwords.
protocol	**socket_server**	**Telnet protocol. SSH (socket_ssh) and raw (raw_data) are also supported.**
ipno	192.168.1.101+	This value can be kept unless you access through IP address is required.
socket_port	**7001+**	**Change it to the TCP port to be used by the first serial port; keep the *Incremented* option on.**
issue	\r\nWelcome to...	This will be the banner when a login is required.
prompt	%h login	Login prompt. %h is the hostname.
term	vt100	Other terminal types are available (ansi, linux).
tx_interval	100	Send buffer received to the application each 100ms.
poll_interval	0	Don’t send modem state commands.

Parameter	Value	Comments
idletimeout	0	Don’t finish the session by idle timeout.
data_buffering	**0**	**Data buffering disabled.**
DB_timestamp	0	Don’t include time in the data buffering.
alarm	**0**	**Don’t generate alarm syslogs.**
syslog_buffering	**0**	**Don’t generate syslogs for data buffering.**
dont_show_DB_ menu	1	Don’t show DB menu when the session is opened. break_sequence "break. this sequence will generate a BREAK in the serial port, in an SSH session.
sniff_mode	**out**	**Only output packets can be traced.**
admin_users	**peter, john**	**If the sniff must be disabled and these users don't exist, this value can be kept.**
multiple_sessions	no	Don’t accept multiple sessions for sniff.
escape_char	^z	This character will cause a sniff session to switch to the command mode.
serverfarm	Server_connected serial<*port_number*>	Name of the SSH server connected to the serial port

TS profile:

Parameter	Value	Comments
speed	**9600**	**Port speed - 9600 bps.**
datasize	8	8 bits.
stopbits	1	1 stop bit.
parity	None	No parity
flow	hard	Hardware flow control.
dcd	1	Sensitive to DCD signal.
sysutmp	1	Write the users in utmp log file.
syswtmp	0	Don't write the users in wtmp log file.

Parameter	Value	Comments
authtype	none	No authentication; the next six parameters will be used only if authentication has radius or tacacs+.
authhost1	200.200.200.2	Change it to the authentication server of your network
accthost1	200.200.200.2	Change it to the accounting server of your network
radtimeout	3	3 minutes of timeout.
secret	black box	Change it to the secret of your RADIUS server.
radretries	5	5 retries before giving up authenticating.
radnullpass	0	Don't allow users with null passwords.
protocol	**telnet**	**The other protocols are login, rlogin, ssh and socket_client.**
socket_port	**23**	**TCP port (22 for ssh, 513 for rlogin).**
host	**200.200.200.3**	**Change it to the server to which the Secure Console Port Server SSH will log in.**
issue	\r\nWelcome to	This will be the banner when a login is required.
prompt	%h login	Login prompt. %h is the hostname.
term	vt100	Other terminal types are available (ansi, linux).

RAS profile:

Parameter	Value	Comments
speed	57600	This speed is normally used for communication with modems.
datasize	8	8 bits character.
stopbits	1	1 stop bit.
parity	none	No parity.
flow	hard	Hardware flow control.
dcd	1	Sensitive to DCD signal.
sysutmp	1	Write the users in utmp log file.
syswtmp	0	Don't write the users in wtmp log file.
authtype	radius	RADIUS authentication; the next six parameters will be used only if authentication has radius or tacacs+.

Parameter	Value	Comments
authhost1	**200.200.200.2**	**Change it to the authentication server of your network.**
accthost1	**200.200.200.2**	**Change it to the accounting server of your network.**
radtimeout	3	3 minutes of timeout.
secret	**black box**	**Change it to the secret of the RADIUS/. TACACS+ server.**
radretries	5	5 retries before giving up authenticating.
radnullpass	0	Don't allow users with null passwords.
protocol	ppp	The other protocols are slip and cslip.
ipno	**200.200.200.11+**	**Change it to the IP address of the remote user connected to the first port. Keep the *Incremented* option on.**
issue	\r\nWelcome to ...	This will be the banner when a login is required.
prompt	%h login	Login prompt. %h is the hostname.
netmask	255.255.255.255	This option are to be changed only for LAN-to-LAN profile.
mtu	1500	This is the default MTU
mru	1500	This is the default MRU
initchat	TIMEOUT 10	This chat script fits for most of the modems
autoppp	%i:%j novj ...	Van Jacobson disabled, ACCM with characters XON/XOFF, IPX disabled, CCP disabled, authentication PAP, callback enabled.
pppopt	%i:%j novj ...	Van Jacobson disabled, ACCM with characters XON/XOFF, IPX disabled, CCP disabled.

Automation profile:

Parameter	Value	Comments
speed	**9600**	**Port speed - 9600bps.**
datasize	8	8 bits character.
stopbits	1	1 stop bit.
parity	none	No parity.

Parameter	Value	Comments
flow	hard	Hardware flow control.
dcd	0	Not sensitive to DCD signal.
media	**rs232**	**Change this option if RS485 media is used.**
modbus_smode	**ascii**	**Change this option if RTU serial mode is used.**
sysutmp	1	Write the users in utmp log file.
syswtmp	0	Don't write the users in wtmp log file.
authtype	none	No authentication.
protocol	modbus	MODBUS protocol.
socket_port	520	MODBUS port.

APPENDIX G GENERATING ALARM AND SYSLOG

Versions 1.3.3 and later

This appendix shows the characteristics of the Alarm for Data Buffering that is implemented for all the Secure Console Port Server SSH family. It is divided in five parts:

1. Syslog-ng and its configuration
2. Alarm, sendmail, sendsms and snmtrap
3. Example of the configuration to use syslog_buffering
4. Example of the configuration to use alarm feature
5. Example of the configuration to use multiples syslog servers.

1. Syslog-ng

The syslog-ng reads from sources (files, TCP/UDP connections, syslogd clients), filters the messages and takes an action(writes in files, sends snmptrap, pager, e-mail or syslogs).

The configuration file is read at startup and is reread after receipt of a hangup (HUP) signal. When reloading the configuration file, all destination files are closed and reopened as appropriate.

You will need to define sources, filters and actions (destinations), and after you'll connect them as explained below.

You can specify several global options to syslog-ng in the options statement:

```
options { opt1(params); opt2(params); ... };
```

where *optn* can be any of the following:

- time_reopen(n): the time to wait before a died connection is reestablished.
- time_reap(n): the time to wait before an idle destination file is closed.
- sync_freq(n): the number of lines buffered before written to file. (the file is synced when this number of messages has been written to it)
- mark_freq(n): the number of seconds between two MARKS lines.
- log_fifo_size(n): the number of lines fitting to the output queue.
- chain_hostname(yes/no) or long_hostname(yes/no):

Enable/disable the chained hostname format.

- use_time_recvd(yes/no): Use the time a message is received instead of the one specified in the message.
- use_dns(yes/no): Enable or disable DNS usage. syslog-ng blocks on DNS queries, so enabling DNS may lead to a Denial of Service attach.
- gc_idle_threshold(n): Sets the threshold value for the garbage collector, when syslog-ng is idle. GC phase starts when the number of allocated objects reach this number. Default: 100.
- gc_busy_threshold(n): Sets the threshold value for the garbage collector, when syslog-ng is busy. GC phase starts
- create_dirs(yes/no): Enable creating no-existing directories.
- owner(name): Set the owner of the created file to the one specified. Default: root
- group(name): Set the group of the created file to the one specified. Default: root
- perm(mask): Set the permission mask of the created file to the one specified. Default: 0600

To define sources use this statement:

source <identifier> { source-driver([params]); source-driver([params]); ...};

where:

identifier: it has to uniquely identify this given source;
source-driver: it is a method of getting a given message.
params: each source-driver may take parameters, some of them required, some of them optional.

The following source-drivers are available:

a) internal()
    - messages generated internally in syslog-ng

b) unix_stream(filename [options]) and unix_dgram(filename [options])
    - they open the given AF_UNIX socket, and start listening on them for messages.
    - options: owner(name), group(name), perm(mask) are equal global options
        keep-alive(yes/no) - selects whether to keep connections opened when syslog-ng is restarted, can be used only with unix_stream. Default: yes
        max-connections(n) - limits the number of simultaneously opened connections. Can be used only with unix_stream. Default: 10.

c) tcp([options]) and udp([options])
    - these drivers let you receive messages from the network, and as the name of the drivers show, you can use both TCP and UDP.
    - none of tcp() and udp() drivers require positional parameters. By default the bind to 0.0.0.0:514, which means that syslog-ng will listen on all available interfaces.
    - options: ip(<ip address>) - the IP address to bind to. Default: 0.0.0.0
        port(<number>) - UDP/TCP port used to listen messages. Default: 514
        max-connections(n) - limits the number of simultaneously opened connections. Default: 10.

d) file(filename)
    - it opens the specified file, and reads messages.

e) pipe(filename)
    - it opens a named pipe with the specified name, and listens for messages. (you'll need to create the pipe using mkfifo command).

Some examples:

1. To read from a file: source <identifier> {file(filename);};

   Example to read messages from “/temp/file1” file:

   ```
   source file1 {file(“/temp/file1”);};
   ```

   Example to receive messages from kernel:

   ```
   source s_kernel { file(“/proc/kmsg”); };
   ```

2. To receive messages from local syslogd clients:

   ```
   source sysl {unix-stream(“/dev/log”);};
   ```

3. To receive messages from remote syslogd clients:

   ```
   source s_udp { udp(ip(<cliente ip>) port(<udp port>)); };
   ```

   Example to listen messages from all machines on UDP port 514:

   ```
   source s_udp { udp(ip(0.0.0.0) port(514));};
   ```

   Example to listen messages from one client (IP address=10.0.0.1) on UDP port 999:

   ```
   source s_udp_10 { udp(ip(10.0.0.1) port(999)); };
   ```

To define filters use this statement:

filter <identifier> { expression; };

where: identifier - has to uniquely identify this given filter

expression - boolean expression using internal functions, which has to evaluate to true for the message to pass.

The following internals functions are available:

a) facility(<facility code>):
   - selects messages based on their facility code.

b) level(<level code>) or priority(<level code>):
   - selects messages based on their priority

c) program(<string>):
   - tries to match the <string> to the program name field of the log message

d) host(<string>):
   - tries to match the <string> to the hostname field of the log message

e) match(<string>):
   - tries to match the <string> to the message itself.

Some examples:

1. To filter by facility: filter f_facilty { facility(<facility name>); };

   Examples:

   ```
   filter f_daemon { facility(daemon); };
   filter f_kern { facility(kern); };
   filter f_debug { not facility(auth, authpriv, news, mail); };
   ```

2. To filter by level: filter f_level { level(<level name>);};

   Examples:

   ```
   filter f_messages { level(info..warn);};
   filter f_emergency { level(emerg); };
   ```

filter f_alert { level(alert); };

3. To filter by matching one string in the received message: filter f_match { match(“string”); };
   Example to filter by matching the string “named”:
   filter f_named { match(“named”); };

4. To filter ALARM messages:
   filter f_alarm { facility(local[0+DB_facility]) and level(info) and match(“ALARM”) and match(“<your string>”); } ;
   Example to filter ALARM message with the string “kernel panic”:
   filter f_kpanic { facility(local1) and level(info) and match(“ALARM”) and match(“kernel panic”); };
   Example to filter ALARM message with the string “root login”:
   filter f_root { facility(local1) and level(info) and match (“ALARM”) and match(“root login”); };

5. Example the filter to eliminate sshd debug messages
   filter f_sshd_debug { not program(“sshd”) or not level(debug); };

6. To filter the syslog_buffering ;
   filter f_syslog_buf { facility(local[0+<conf.DB_facility>]); };

To define actions use this statement:

destination <identifier> { destination-driver([params]); destination-driver([param]); ..};

where: identifier - has to uniquely identify this given destination.
destination-driver: it is a method of outputing a given message.
params: each destination-driver may take parameters, some of them required, some of them optional.

The following destination drivers are available:

a) file(filename [options])

- this is one of the most important destination drivers in syslog-ng. It allows you to output log messages to the named file.

- the destination filename may include macros (by prefixing the macro name with a '$' sign) which gets expanded when the message is written.

- since the state of each created file must be tracked by syslog-ng, it consumes some memory for each file. If no new messages are written to a file within 60 seconds (controlled by the time_reap global option), it's closed, and its state is freed.

- available macros in filename expansion:

- HOST - the name of the source host where the message is originated from.
- FACILITY - the name of the facility, the message is tagged as coming from.
- PRIORITY or LEVEL - the priority of the message
- PROGRAM - the name of the program the message was sent by.
- YEAR, MONTH, DAY, HOUR, MIN, SEC - the year, month, day, hour, min, sec of the message was sent.
- TAG - it equal FACILITY/LEVEL
- FULLHOST - the name of the source host and the source-driver: <source-driver>@<hostname>
- MSG or MESSAGE - the message received.
- FULLDATE - the date of the message was sent.

- available options:

- log_fifo_size(number) - the number of entries in the output file.
- sync_freq(number) - the file is synced when this number of messages has been written to it.
- encrypt(yes/no) - encrypt the resulting file.
- compress(yes/no) - compress the resulting file using zlib.
- owner(name), group(name), perm(mask) - equal global options
- template("string") - syslog-ng write the "string" in the file.

b) pipe(filename [options])

- this driver sends messages to a named pipe.

- available options:

- owner(name), group(name), perm(mask) - equal global options

- template(“string”) - syslog-ng write the “string” in the file. You can use the MACROS in the string.

c) unix-stream(filename) and unix-dgram(filename)
- this driver sends messages to a unix socket en either SOCKET_STREAM or SOCK_DGRAM mode.

d) udp (“<ip address>” port(number)) and tcp (“<ip address>” port(number))
- this driver sends messages to another host (ip address/port) using either UDP or TCP protocol.

e) usertty(<username>)
- this driver writes messages to the terminal of a logged-in username.

f) program(<program name and arguments>)
- this driver fork()’s executes the given program with the arguments and sends messages down to the stdin of the child.

Some examples:

1. To send e-mail:

```
        destination <ident> { pipe(“/dev/cyc_alarm” template(“sendmail <pars>”));};
    where ident: uniquely identify this destination
              pars: -t <name>[,<name>]: To address
                    [-c <name>[,<name>]]: CC address
                    [-b <name>[,<name>]]: Bcc address
                    [-r <name>[,<name>]]: Reply-to address
                    -f <name>: From address
                    -s \”<text>\”: Subject
                    -m \”<text message>\”: Message
                    -h <IP address or name>: SMTP server
                    [-p <port>]: port used. default: 25
```

To mount the message, use this macros:

$FULLDATE - the complete date when the message was sent.
$FACILITY - the facility of the message
$PRIORITY or $LEVEL - the priority of the message
$PROGRAM - the message was sent by this program (BUFFERING or SOCK)
$HOST - the name of the source host.
$FULLHOST - the name of the source host and the source driver. Format: <source>@<hostname>
$MSG or $MESSAGE - the message received

Example to send e-mail to z@none.com (SMTP's IP address 10.0.0.2) from the e-mail address a@none.com with subject "ALARM". The message will carry the currante date, the hostname of this Secure Console Port Server SSH and the message that was received from the source.

```
destination d_mail1 {
        pipe("/dev/cyc_alarm"
                template("sendmail -t z@none.com -f a@none.com -s \"ALARM\" \ -m \"$FULLDATE $HOST $MSG\" -h 10.0.0.2"));
};
```

2. To send to pager server (sms server):

```
destination <ident> {pipe("/dev/cyc_alarm" template("sendsms <pars>"));};
```

where ident: uniquely identify this destination

```
pars : -d <mobile phone number>
       -m \"<message - max.size 160 characters>\"
       -u <username to login on sms server>
       -p <port sms - default: 6701>
       <server IP address or name>
```

Example to send a pager to phone number 123 (Pager server at 10.0.0.1) with message carrying the current date, the hostname of this Secure Console Port Server SSH and the message that was received from the source:

```
destination d_pager {
```

```
                pipe(“/dev/cyc_alarm”
                template(“sendsms -d 123 -m \”$FULLDATE $HOST $MSG\” 10.0.0.1"));
        };
```

3. To send snmptrap:

```
              destination <ident> {pipe(“/dev/cyc_alarm” template(“snmptrap <pars>”)); };
        where ident: uniquely identify this destination
              pars: -v 1
                     <snmptrapd IP address>
                     public                    : community
                     \”\”                      : enterprise-oid
                     \”\”                      : agent/hostname
                     <trap number>             : 2-Link Down, 3-Link Up, 4-Authentication Failure
                     0                         :
                     \”\”                      : host-uptime
                     .1.3.6.1.2.1.2.2.1.2.1    :interfaces.iftable.ifentry.ifdescr.1
                     s                         : string
                     \”<message - max. size 250 characters>\”
```

Example to send a Link Down trap to server at 10.0.0.1 with message carrying the current date, the hostname of this Secure Console Server SSH and the message that received from the source:

```
              destination d_trap {
                     pipe(“/dev/cyc_alarm”
                      template(“snmptrap -v 1 10.0.0.1 public \”\” \”\” 2 0 \”\” \  .1.3.6.1.2.1.2.2.1.2.1 s
\”$FULLDATE $HOST $MSG\” “));
              };
```

4. To write in file:

```
              destination d_file { file(<filename>);};
```

Example send message to console:

```
                destination d_console { file(“/dev/ttyS0”);};
```

Example write message in /var/log/messages file:

```
                destination d_message { file(“/var/log/messages”); };
```

5. To write messages to the session of logged-in user:
   ```
   destination d_user { usertty("<username>"); };
   ```
   Example to send message to all sessions with root user logged:
   ```
   destination d_userroot { usertty("root"); };
   ```
6. To send message to remote syslogd server:
   ```
   destination d_udp { udp( "<remote IP address>" port(514)); };
   ```
   Example to send syslogs to syslogd located at 10.0.0.1:
   ```
   destination d_udp1 { udp( "10.0.0.1" port(514)); };
   ```

To connect the sources, filters and actions (any message coming from one of the listed sources, matching the filters (each of them) is sent to the listed destinations). Use this statement:

```
log { source(S1); source(S2); ...
      filter(F1);filter(F2);...
      destination(D1); destination(D2);...
};
```

where: Sx - identifier of the sources defined before
Fx - identifier of the filters defined before
Dx - identifier of the actions/destinations defined before

Examples:

1. To send all messages received from local syslog clients to console
   ```
   log { source(sysl); destination(d_console);};
   ```

2. To send only messages with level alert and received from local syslog clients to all logged root user:
   ```
   log { source(sysl); filter(f_alert); destination(d_userroot); };
   ```

3. To writes all messages with levels info, notice or warning and received from syslog clients (local and

remotes) to /var/log/messages file:

```
log { source(sysl); source(s_udp); filter(f_messages); destination(d_messages); };
```

4. To send e-mail if message received from local syslog client has the string “kernel panic”:

```
log { source(sysl); filter(f_kpanic); destination(d_mail1); };
```

5. To send e-mail and pager if message received from local syslog client has the string “root login”:

```
log { source(sysl); filter(f_root); destination(d_mail1); destination(d_pager); };
```

6. To send messages with facility kernel and received from syslog clients (local and remote) to remote syslogd:

```
log { source(sysl); source(s_udp); filter(f_kern); destination(d-udp1); };
```

2. Alarm, Sendmail, Sendsms and Snmptrap

2.1. Alarm

This feature is available only in the Console Server Application.

Secure Console Port Server SSH sends messages using pager, e-mail or snmptrap if the serial port receives message with specific string.

To configure this feature you will need:

1. to activate alarm in Portslave configuration file (parameter all.alarm - 0 inactive or <> 0 active)

2. to configure filters in syslog-ng configuration file:

```
filter f_alarm { facility(local[0+DB_facility]) and level(info) and match(“ALARM”) and match(“<your string>”);
} ;
```

For example, to filter ALARM message with the string “kernel panic”:

```
filter f_kpanic { facility(local1) and level(info) and match(“ALARM”) and match(“kernel panic”); };
```

For example, to filter ALARM message with the string “root login”:

```
filter f_root { facility(local1) and level(info) and match(“ALARM”) and match(“root login”); };
```

3. to configure actions in syslog-ng configuration file.

   Examples (see more details in syslog-ng examples):
   to send e-mail: destination d_mail { pipe(“/dev/cyc_alarm” template(“sendmail <pars>”));};
   to send pager: destination d_pager {pipe(“/dev/cyc_alarm” template(“sendsms <pars>”));};
   to send snmptrap: destination d_trap {pipe(“/dev/cyc_alarm” template(“snmptrap <pars>”)); };

4. to connect filters and actions in syslog-ng configuration file.

   Example: alarm is active and if the serial port receives the string “kernel panic”, one message will be sent to the pager.
   log (source(sysl); filter(f_kpanic); destination(d_trap); destination(d_pager); };

2.2. Sendmail

Sendmail sends a message to a SMTP server. It is not intended as a user interface routine; it is used only to send pre formatted messages. Sendmail reads all parameters in command line.

If the SMTP server does not answer the SMTP protocol requests sent by sendmail, the message is dropped.

Synopsis: sendmail -t <name>[,<name>] [-c <name> [,<name>]] [-b <name> [,<name>]] [-r <name>] -f <name> -s <text> -m <text> -h <SMTP server> [-p <smtp-port>]

where:

-t <name>[,<name>]

"To: ". Required. Multi-part allowed (multiple names are separated by commas). Names are expanded as explained below.

[-c <name> [,<name>]]

"Cc: ". Optional. Multi-part allowed (multiple names are separated by commas).

[-b <name> [,<name>]]

"Bcc:". Optional. Multi-part allowed (multiple names are separated by commas).

[-r <name>]

"Reply-To: ". Optional. Use the Reply-To: field to make sure the destination user can send a reply to a regular mailbox.

-f <name>

"From: " Required.

-s <text>

"Subject: ". Required.

-m <text>

"body". The message body.

-h <SMTP server>

Required. IP address or name of the SMTP server.

[-p <SMTP port>

Optional. The port number used in the connection with the server. Default: 25.

<name>: Any email address.

<text>: A text field. As this kind of field can contain blank spaces, please use the quotation marks to enclose the text.

For example, to send e-mail to z@none.com (SMTP's IP address 10.0.0.2) from the e-mail address a@none.com with subject "Console Server sendmail test ".

sendmail -t z@none.com -f a@none.com -s "Console Server sendmail test" -m "Sendmail test. \n Is it OK??? " -h 10.0.0.2

2.3. Sendsms

The sendsms is the Linux command line client for the SMSLink project (Philippe Andersson - "Les Ateliers du Heron"). It accepts command line parameters that define the message to be sent, and transmits them to the SMS server process running on the designated server. The sendsms was developed specifically for easy calling from shell scripts or similar situations.

Synopsis: sendsms [-r] [-g] [-v] -d dest (-m message or -f msgfile) [-u user] [-p port] server, where:

-r : Reporting. Additional info will be included in the message printed on stderr (namely, the device name used by the server to send the SMS out, and the message ID attributed to the SMS by the module's SIM card). If any of these items is missing or can't be parsed, a value of "??" will be returned.

-g : Turns debugging on. Will output the entire dialog with the server on stderr (and more).

-h : Displays a short help message and exits.

-v : Displays version information and exits.

-d dest: Required. The GSM network address (i.e. phone number) of the mobile phone the message is to be sent to. Supported format is: [int. prefix - country code] area code - phone number.The international prefix can be either “+” or “00” (or any other value supported by the GSM network provider the server is subscribed to). Some separation characters can be used to beautify the number, but they are purely cosmetic and will be stripped by the server. Those characters are [./-]. The pause character (‘,’) is not supported. Regarding the international country code, don’t forget that its necessity is to be considered respective to the SMS gateway location (the host this client program is connecting to), not the location where the client is run from. In case of doubt, please contact the SMS server administrator for your network. Please always include the area code (even when sending to a destination in the same “area”, i.e. on the same network). The number without the area code, though syntactically correct and accepted by the network, would never get delivered (at least, that’s my experience with Proximus — YMMV).

-m message : Required (Use one and only one of “-m” or “-f”). The text of the message to be sent. Unless made up of a single word, it will have to be quoted for obvious reasons. Maximum length is 160 characters. A longer message will be truncated (the user will be warned about it), but the message will still be sent. At the present time, only 7bit ASCII is supported for the message text.

-f msgfile : Required (use one and only one of “-m” or “-f”). The name of a text file where the message to send is to be read from. This file can contain multiple lines of text (they will be concatenated), but its total length can’t exceed 160 characters. A longer text will be truncated (the user will be warned about it), but the message will still be sent. The special file ‘-’ means that input will be read from stdin. At the present time, only 7bit ASCII is supported for the message text.

-u user : Optional. The server module requires the user to identify himself for logging purposes.No authentication is performed on this info though. If this parameter is omitted, sendsms will send the Unix username of the current user. This parameter allows you to override this default behavior (might be useful in the case of automated sending).

-p port : Optional. Communication port on the target server. If provided here, this value will be used to connect to the server. If omitted, the client will query the local system for the port number associated with the "well known service" sms (as defined in /etc/services). If that doesn't return an answer, the compiled-in default value 6701 will be used.

server : Required. The host name or IP address of the computer where the SMS gateway server process is running. By default, this server will be listening on TCP port 6701.

Upon success (when the server module reports that the message was successfully sent), sendsms returns 0. When a problem occurs, a non zero value is returned. Different return values indicate different problems. A return value of 1 indicates a general failure of the client program.

COPYRIGHT: SMSLink is (c) Les Ateliers du Heron, 1998 by Philippe Andersson <philipa@tiscalinet.be>. It has been originally written for Scitex Europe, S.A. Part of the code is (c) Riccardo Facchetti. The code also includes contributions from Philipp Klaus <pklaus@access.ch> and numerous others. All contributors are acknowledged in the CHANGELOG document, and in the comment headers of the source files they modified. SMSLink has been released to the public under the GNU GPL.

Example to send a pager to phone number 123 (Pager server at 10.0.0.1) with message:

sendsms -d 123 -m "Hi, this is a test message send from Secure COnsole Port Server SSH using sendsms" 10.0.0.1

2.4. Snmptrap

Snmptrap is an SNMP application that uses the TRAP-PDU Request to send information to a network manager. One or more fully qualified object identifiers can be given as arguments on the command line. A type and a value must accompany each object identifier. Each variable name is given in the format specified. If any of the required version 1 parameters: enterprise-oid, agent and uptime are specified as empty, it defaults to ".1.3.6.1.4.1.3.1.1", hostname and host-uptime respectively.

Synopsis:

snmptrap -v 1 [-Ci] [common arguments] enterprise-oid agent generic-trap specific-trap uptime [objectID type value]...

snmptrap -v [2c|3] [-Ci] [common arguments] uptime trap-oid [objectID type value]...

where:

-Ci : Optional. It sends INFORM-PDU

common arguments: required. They are: SNMP server IP address and community.

enterprise-oid: required, but it can be empty (‘’).

agent: required, but it can be empty(‘’). The agent name.

generic-trap: required. The generic trap number: 2 (link down), 3 (link up), 4 (authentication failure), ...

specific-trap: required. The specific trap number.

uptime: required.

[objectID type value]: optional. objectID is the object oid, you want to inform its value to server.

If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint in what way the request was malformed. If there were other variables in the request, the request will be resent without the bad variable.

For example, to send a Link Down trap to server at 10.0.0.1 with interfaces.iftable.ifentry.ifdescr :

```
snmptrap -v 1 10.0.0.1 public "" 2 0  ""  .1.3.6.1.2.1.2.2.1.2.1 s "Secure Console Port Server SSH: serial port number 1 is down"
```

3. Syslog-ng configuration to use with syslog buffering feature

This configuration example is to use syslog buffering feature, and to send the messages to remote syslogd (10.0.0.1).

In the pslave.conf file the parameters of the syslog buffering feature are configured as:

```
conf.DB_facility        1
all.syslog_buffering    100
```

The syslog-ng.conf file need these lines:

```
# local syslog clients
source src { unix-stream("/dev/log"); };

destination d_buffering { udp("10.0.0.1")); };

filter f_buffering { facility(local1) and level(notice); };

# send only syslog_buffering messages to remote server
log { source(src); filter(f_buffering); destination(d_buffering); };
```

4. Syslog-ng configuration to use with alarm feature

This configuration example is to use alarm feature.

In the pslave.conf file the parameters of the alarm feature are configured as:

```
all.alarm         1
conf.DB_facility        2
```

The syslog-ng.conf file need these lines:

```
# local syslog clients
source src { unix-stream(“/dev/log”); };

# To filter ALARM message with the string “kernel panic”:
filter f_kpanic { facility(local2) and level(info) and match(“ALARM”) and match(“kernel panic”); };

# To filter ALARM message with the string “root login”:
 filter f_root { facility(local1) and level(info) and match(“ALARM”) and match(“root login”); };

# To send e-mail to z@none.com (SMTP’s IP address 10.0.0.2)

# from the e-mail address a@none.com with subject “ALARM”.

# The message will carry the current date, the hostname

# of this Secure Console Port Server SSH and the message that was received from the source.
destination d_mail1 {
        pipe(“/dev/cyc_alarm”
           template(“sendmail -t z@none.com -f a@none.com -s \”ALARM\” \  -m \”$FULLDATE $HOST $MSG\”
   -h 10.0.0.2"));
          };

# Example to send a pager to phone number 123 (Pager server at 10.0.0.1) with message
# carrying the current date, the hostname of this Secure Console Port Server SSH and the message that was
   received from the source:
destination d_pager {
          pipe("/dev/cyc_alarm"
                template("sendsms -d 123 -m \"$FULLDATE $HOST $MSG\" 10.0.0.1"););
};

# Example to send a Link Down trap to server at 10.0.0.1 with message carrying the current

# date, the hostname of this Secure Console Port Server SSH and the message that received from the
   source:
destination d_trap {
       pipe("/dev/cyc_alarm"
             template("snmptrap -v 1 10.0.0.1 public \"\" \"\" 2 0 \"\" \
                 .1.3.6.1.2.1.2.2.1.2.1 s \"$FULLDATE $HOST $MSG\" ");););
};

# To send e-mail and snmptrap if message received from local syslog client has the string "kernel panic":
log { source(sysl); filter(f_kpanic); destination(d_mail1); destination(d_trap); };

# To send e-mail and pager if message received from local syslog client has the string
# "root login":
log { source(sysl); filter(f_root); destination(d_mail1); destination(d_pager); };
```

5. Syslog-ng configuration to use with multiple remote syslog servers

This configuration example is to use multiple remote syslog servers.

In the pslave.conf file the facility parameter is configured as:

```
conf.facility      1
```

The syslog-ng.conf file need these lines:

```
# local syslog clients
source src { unix-stream(“/dev/log”); };

# remote server 1 - IP address 10.0.0.1 port default
destination d_udp1 { udp(“10.0.0.1”);};

# remote server 2 - IP address 10.0.0.2 port 1999
destination d_udp2 { udp(“10.0.0.2” port(1999);};

# filter messages from facility local1 and level  info to warning
filter f_local1 { facility(local1) and level(info..warn)};

# filter messages from facility local 1 and level err to alert
filter f_critic { facility(local1) and level(err .. alert)};

# send info, notice and warning messages to remote server udp1
log { source(src); filter(f_local1); destination(d_udp1); };

# send error, critical and alert messages to remote server udp2
log { source(src); filter(f_critic); destination(d_udp2); };
```

APPENDIX H CERTIFICATE FOR HTTP SECURITY

Obtaining a Signed Digital Certificate

A certificate for the HTTP security is created by a CA (Certification Authority). The most usual procedure to obtain a certificate is:

- Generation of the public and private keys, using a public key algorithm like RSA or X509. The keys can be generated by using a key generator software. In a Linux computer, this can be done using the OpenSSL package, through the following command:

```
# openssl req -new -nodes -keyout private.key -out public.csr
```

If this command is used, the following information is required:

Parameter	Description
Country Name (2 letters code) [AU]:	The country code with two letters.
State or Province Name (full name) [Some-State]:	Provide the full name (not the code) of the state.
Locality Name (e.g., city) []:	Enter the name of your city.
Organization Name (e.g., company) [Internet Widgits Pty Ltd]:	Organization that you work or want to certificate for.
Organizational Unit Name (e.g., section) []:	Department or section which you work.
Common Name (e.g., your name or your server's hostname) []:	Name of the machine where the certificate must be installed.
Email Address []:	Your email address or the administrator email address.

The other requested information can be skipped.

- The certificate signing request (CSR) generated by the command above contains some personal (or corporate) information and its public key. The next step is to submit the CSR and some personal data to the CA. This service can be requested by accessing the CA website and is not free, and there is a list of CA's in the URL http://www.pki-page.org/.

- The request will be analyzed by the CA, for policy approval and to be signed.

- After the approval, the CA will send a certificate file to the origin, which we will call Cert.cer, for example purposes. The certificate is also stored on a directory server.

- The certificate must be installed in the GoAhead web server, by following these instructions:

1. Open a Black Box Terminal Server session and do the login.

2. Load the files Cert.cer (certificate file received from the CA) and private.key (private key generated by openssl) to a temporary directory (/tmp), using FTP or NFS service.

3. Join the certificate with the private key into the file /web/server.pem.

```
#cat Cert.cer private.key > /web/server.pem
```

4. Copy the certificate to the file /web/cert.pem

```
#cp Cert.cer /web/cert.pem
```

5. Include the files /web/server.pem and /web/cert.pem in /etc/config_files.

6. Save the configuration in flash.

```
#saveconf
```

7. The certification will be effective in the next reboot.

APPENDIX I USING MODBUS PROTOCOL IN CAS PROFILE

MODBUS is an application layer messaging protocol for client/server communication which is widely used in the industrial automation. It is a confirmed service protocol and offers many services specified by function codes, like reading and writing registers on PLCs.

A protocol converter for the MODBUS protocol over the TCP/IP communication stack (Modbus/TCP) is implemented in Secure Console Port Server SSH and converts Modbus/TCP ADUs from the Ethernet interface to plain MODBUS message frames over a serial RS-232 or RS-485 interface, and vice-versa, supporting both serial modes (ASCII and RTU).

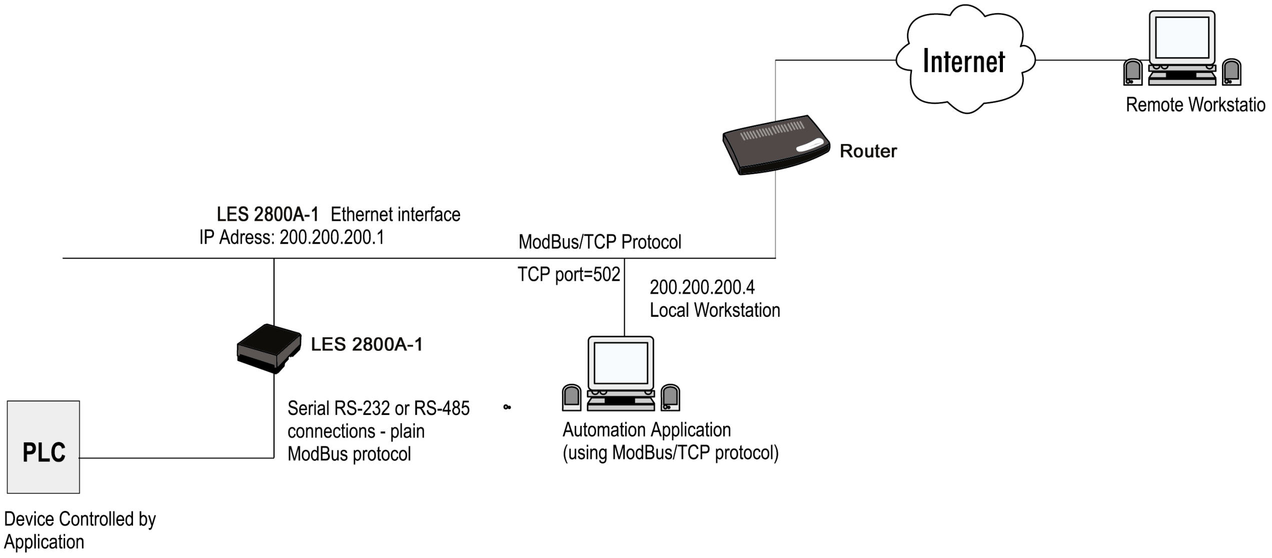


FIGURE I.1 - MODBUS APPLICATION

In this example, the Automation Application running in the Workstation (local or remote) controls the PLCs connected to the serial port (RS-485) of the Secure Console Port Server SSH 1-Port using MODBUS/TCP protocol. The connection is opened using Secure Console Port Server SSH 1-Port Ethernet IP address and TCP port = 502. Secure Console Port Server SSH 1-Port accepts the incoming connection and converts MODBUS/ TCP ADUs (packets) to plain MODBUS frames and sends them over the serial port. On the other hand, the MODBUS frames received from the serial port are converted to MODBUS/TCP ADUs and sent through the TCP connection to the Automation Application.

The configuration described earlier for Console Access Servers (see Chapter 6) should be followed with the following exceptions for this example:

Parameter	**Description**	**Value for This Example**
all.authtype	There are several authentication type options: local (authentication is performed using the /etc/passwd file), radius (authentication is performed using a Radius authentication server), TacacsPlus (authentication is performed using a TacacsPlus authentication server), none, local/radius (authentication is performed locally first, switching to Radius if unsuccessful), radius/local (the opposite of the previous option), RadiusDownLocal (local authentication is tried only when the Radius server is down), local/TacacsPlus (authentication is performed locally first, switching to TacacsPlus if unsuccessful), TacacsPlus/local (the opposite of the previous option), TacacsPlusDownLocal (local authentication is tried only when the TacacsPlus server is down). Note that this parameter controls the authentication required by the Secure Console Port Server SSH. The authentication required by the device to which the user is connecting is controlled separately.	none

FIGURE I.2 - MODBUS PSLAVE.CONF PORT SPECIFIC PARAMETERS

(ONLY WHERE IT DIFFERS FROM THE STANDARD CAS PROFILE)

Parameter	Description	Value for This Example
all.protocol	For the console server profile, the possible protocols are socket_server (when telnet is used), socket_ssh (when ssh version one or two is used), raw_data (to exchange data in transparent mode – similar to socket_server mode, but without telnet negotiation, breaks to serial ports, etc.), or modbus (an application layer messaging protocol for clent/server communication widely used for industrial automation).	modbus
all.modbus_smode	Communication mode through the serial ports. This parameter is meaningful only when modbus protocol is configured. The valid options are ascii (normal TX/RX mode) and rtu (some time constraints are observed between characteres while transmiting a frame). If not configured, ASCII mode will be assumed.	ascii

FIGURE I.2 - MODBUS PSLAVE.CONF PORT SPECIFIC PARAMETERS (CONT.)

(ONLY WHERE IT DIFFERS FROM THE STANDARD CAS PROFILE)

Note: The MODBUS port can be configured in the file /etc/services, changing the corresponding line. By default, the port is 502, as specified in Modbus/TCP draft to the IETF.

APPENDIX J LINUX-PAM

Overview

Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.

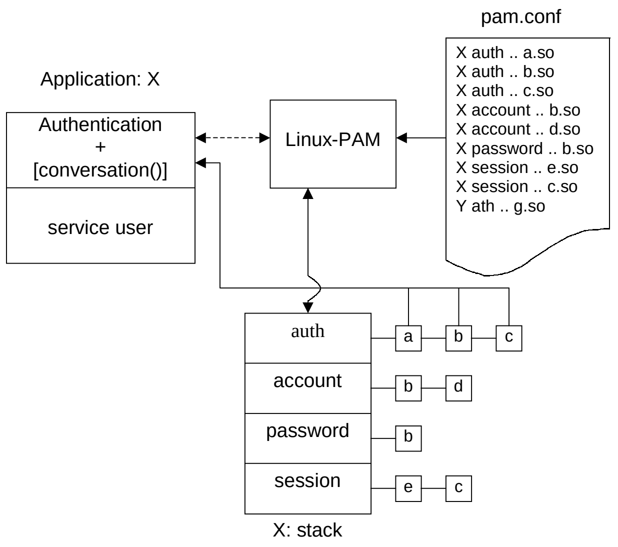
In other words, without (rewriting and) recompiling a PAM-aware application, it is possible to switch between the authentication mechanism(s) it uses. Indeed, one may entirely upgrade the local authentication system without touching the applications themselves.

It is the purpose of the Linux-PAM project to separate the development of privilege granting software from the development of secure and appropriate authentication schemes. This is accomplished by providing a library of functions that an application may use to request that a user be authenticated. This PAM library is configured locally with a system file, /etc/pam.conf (or a series of configuration files located in /etc/pam.d/) to authenticate a user request via the locally available authentication modules. The modules themselves will usually be located in the directory /lib/security and take the form of dynamically loadable object files.

The Linux-PAM authentication mechanism gives to the system administrator the freedom to stipulate which authentication scheme is to be used. He has the freedom to set the scheme for any/all PAM-aware applications on your Linux system. That is, he can authenticate from anything as naive as simple trust (pam_permit) to something as paranoid as a combination of a retinal scan, a voice print and a one-time password!

Linux-PAM deals with four separate types of (management) task. These are: authentication management; account management; session management; and password management. The association of the preferred management scheme with the behavior of an application is made with entries in the relevant Linux-PAM configuration file. The management functions are performed by modules specified in the configuration file.

Here is a figure that describes the overall organization of Linux-PAM.



By way of explanation, the left of the figure represents the application; application X. Such an application interfaces with the Linux-PAM library and knows none of the specifics of its configured authentication method. The Linux-PAM library (in the center) consults the contents of the PAM configuration file and loads the modules that are appropriate for application-X. These modules fall into one of four management groups (lower-center) and are stacked in the order they appear in the configuration file. These modules, when called by Linux-PAM, perform the various authentication tasks for the application. Textual information, required from/or offered to the user, can be exchanged through the use of the application-supplied conversation function.

The Linux-PAM Configuration File

Linux-PAM is designed to provide the system administrator with a great deal of flexibility in configuring the privilege granting applications of their system. The local configuration of those aspects of system security controlled by Linux-PAM is contained in one of two places: either the single system file, /etc/pam.conf; or the /etc/pam.d/ directory. In this section we discuss the correct syntax of and generic options respected by entries to these files.

Configuration file syntax

The reader should note that the Linux-PAM specific tokens in this file are case insensitive. The module paths, however, are case sensitive since they indicate a file's name and reflect the case dependence of typical Linux file-systems. The case-sensitivity of the arguments to any given module is defined for each module in turn.

In addition to the lines described below, there are two special characters provided for the convenience of the system administrator: comments are preceded by a '#' and extend to the next end-of-line; also, module specification lines may be extended with a `\' escaped new-line.

A general configuration line of the /etc/pam.conf file has the following form:

```
Service-name   module-type   control-flag   module-path   arguments
```

Below, we explain the meaning of each of these tokens. The second (and more recently adopted) way of configuring Linux-PAM is via the contents of the /etc/pam.d/ directory. Once we have explained the meaning of the above tokens, we will describe this method.

Service-name

The name of the service associated with this entry. Frequently the service name is the conventional name of the given application. For example, ‘ftpd’, ‘rlogind’, ‘su’, etc.

There is a special service-name, reserved for defining a default authentication mechanism. It has the name ‘OTHER’ and may be specified in either lower or upper case characters. Note, when there is a module specified for a named service, the ‘OTHER’ entries are ignored.

Module-type

One of (currently) the four types of module. The four types are as follows:

Auth - this module type provides two aspects of authenticating the user. Firstly, it establishes that the user is who they claim to be, by instructing the application to prompt the user for a password or other means of identification. Secondly, the module can grant group membership, independently of the /etc/groups, or other privileges through its credential granting properties.

Account -this module performs non-authentication based account management. It is typically used to restrict/ permit access to a service based on the time of day, currently available system resources (maximum number of users) or perhaps the location of the applicant user—‘root’ login only on the console.

Session - primarily, this module is associated with doing things that need to be done for the user before/after they can be given service. Such things include the logging of information concerning the opening/closing of some data exchange with a user, mounting directories, etc.

Password - this last module type is required for updating the authentication token associated with the user. Typically, there is one module for each ‘challenge/response’ based authentication (auth) module-type.

Control-flag

The control-flag is used to indicate how the PAM library will react to the success or failure of the module it is associated with. Since modules can be stacked (modules of the same type execute in series, one after another), the control-flags determine the relative importance of each module. The application is not made aware of the individual success or failure of modules listed in the '/etc/pam.conf' file. Instead, it receives a summary success or fail responses from the Linux-PAM library. The order of execution of these modules is that of the entries in the /etc/pam.conf file; earlier entries are executed before later ones. The control-flag can be defined with one of two syntaxes. The simpler (and historical) syntax for the control-flag is a single keyword defined to indicate the severity of concern associated with the success or failure of a specific module. There are four such keywords: required, requisite, sufficient and optional.

The Linux-PAM library interprets these keywords in the following manner:

Required - this indicates that the success of the module is required for the module-type facility to succeed. Failure of this module will not be apparent to the user until all of the remaining modules (of the same module-type) have been executed.

Requisite - like required, however, in the case that such a module returns a failure, control is directly returned to the application. The return value is that associated with the first required or requisite module to fail. Note, this flag can be used to protect against the possibility of a user getting the opportunity to enter a password over an unsafe medium. It is conceivable that such behavior might inform an attacker of valid accounts on a system. This possibility should be weighed against the not insignificant concerns of exposing a sensitive password in a hostile environment.

Sufficient - the success of this module is deemed 'sufficient' to satisfy the Linux-PAM library that this module-type has succeeded in its purpose. In the event that no previous required module has failed, no more 'stacked' modules of this type are invoked. (Note, in this case subsequent required modules are not invoked.). A failure of this module is not deemed as fatal to satisfying the application that this module-type has succeeded.

Optional - as its name suggests, this control-flag marks the module as not being critical to the success or failure of the user’s application for service. In general, Linux-PAM ignores such a module when determining if the module stack will succeed or fail. However, in the absence of any definite successes or failures of previous or subsequent stacked modules this module will determine the nature of the response to the application. One example of this latter case, is when the other modules return something like PAM_IGNORE.

Newest Syntax

The more elaborate (newer) syntax is much more specific and gives the administrator a great deal of control over how the user is authenticated. This form of the control flag is delimited with square brackets and consists of a series of value=action tokens:

[value1=action1 value2=action2 ...]

Here, valueI is one of the following return values: success; open_err; symbol_err; service_err; system_err; buf_err; perm_denied; auth_err; cred_insufficient; authinfo_unavail; user_unknown; maxtries; new_authtok_reqd; acct_expired; session_err; cred_unavail; cred_expired; cred_err; no_module_data; conv_err; authtok_err; authtok_recover_err; authtok_lock_busy; authtok_disable_aging; try_again; ignore; abort; authtok_expired; module_unknown; bad_item; and default. The last of these (default) can be used to set the action for those return values that are not explicitly defined.

The action can be a positive integer or one of the following tokens: ignore; ok; done; bad; die; and reset.

A positive integer - when specified as the action, can be used to indicate that the next J modules of the current type will be skipped. In this way, the administrator can develop a moderately sophisticated stack of modules with a number of different paths of execution. Which path is taken can be determined by the reactions of individual modules.

Ignore - when used with a stack of modules, the module’s return status will not contribute to the return code the application obtains.

Bad - this action indicates that the return code should be thought of as indicative of the module failing. If this module is the first in the stack to fail, its status value will be used for that of the whole stack.

Die - equivalent to bad with the side effect of terminating the module stack and PAM immediately returning to the application.

Ok - this tells PAM that the administrator thinks this return code should contribute directly to the return code of the full stack of modules. In other words, if the former state of the stack would lead to a return of PAM_SUCCESS, the module's return code will override this value. Note, if the former state of the stack holds some value that is an indicative of a module failure, this 'ok' value will not be used to override that value.

Done - equivalent to ok with the side effect of terminating the module stack and PAM immediately returning to the application.

Reset - clear all memory of the state of the module stack and start again with the next stacked module.

Module-path

The path-name of the dynamically loadable object file, the pluggable module itself, If the first character of the module path is '/', it is assumed to be a complete path. If this is not the case, the given module path is appended to the default module path: /lib/security.

Currently the Secure Console Port Server SSH has the following modules available:

pam_access - Provides logdaemon style login access control.

pam_deny - Deny access to all users.

pam_env - This module allows the (un)setting of environment variables. Supported is the use of previously set environment variables as well as PAM_ITEMs such as PAM_RHOST.

pam_filter - This module was written to offer a plug-in alternative to programs like ttysnoop (XXX - need a reference). Since writing a filter that performs this function has not occurred, it is currently only a toy. The single filter provided with the module simply transposes upper and lower case letters in the input and output streams. (This can be very annoying and is not kind to termcap based editors).

pam_group - This module provides group-settings based on the user's name and the terminal they are requesting a given service from. It takes note of the time of day.

pam_issue - This module presents the issue file (/etc/issue by default) when prompting for a username.

pam_lastlog - This session module maintains the /var/log/lastlog file. Adding an open entry when called via the pam_open_seesion()function and completing it when pam_close_session() is called. This module can also display a line of information about the last login of the user. If an application already performs these tasks, it is not necessary to use this module.

pam_limits - This module, through the Linux-PAM open-session hook, sets limits on the system resources that can be obtained in a user-session. Its actions are dictated more explicitly through the configuration file discussed below.

pam_listfile - The list-file module provides a way to deny or allow services based on an arbitrary file.

pam_motd - This module outputs the motd file (/etc/motd by default) upon successful login.

pam_nologin - Provides standard Unix nologin authentication.

pam_permit - This module is very dangerous. It should be used with extreme caution. Its action is always to permit access. It does nothing else.

pam_radius – Provides Radius server authentication and accounting.

pam_rootok - This module is for use in situations where the superuser wishes to gain access to a service without having to enter a password.

pam_securetty - Provides standard Unix securetty checking.

pam_time - Running a well regulated system occasionally involves restricting access to certain services in a selective manner. This module offers some time control for access to services offered by a system. Its actions are determined with a configuration file. This module can be configured to deny access to (individual) users based on their name, the time of day, the day of week, the service they are applying for and their terminal from which they are making their request.

pam_tacplus – Provides Tacacs+ Server authentication, authorization (account management), and accounting (session management)

pam_unix - This is the standard Unix authentication module. It uses standard calls from the system’s libraries to retrieve and set account information as well as authentication. Usually this is obtained from the etc/passwd and the /etc/shadow file as well if shadow is enabled.

pam_warn - This module is principally for logging information about a proposed authentication or application to update a password.

pam_wheel - Only permit root authentication to members of wheel group.

Arguments

The arguments are a list of tokens that are passed to the module when it is invoked. They are much like arguments to a typical Linux shell command. Generally, valid arguments are optional and are specific to any given module. Invalid arguments are ignored by a module, however, when encountering an invalid argument, the module is required to write an error to syslog(3).

The following are optional arguments which are likely to be understood by any module. Arguments (including these) are in general optional.

debug - Use the syslog(3) call to log debugging information to the system log files.

no_warn - Instruct module to not give warning messages to the application.

use_first_pass - The module should not prompt the user for a password. Instead, it should obtain the previously typed password (from the preceding auth module), and use that. If that doesn't work, then the user will not be authenticated. (This option is intended for auth and password modules only).

try_first_pass - The module should attempt authentication with the previously typed password (from the preceding auth module). If that doesn't work, then the user is prompted for a password. (This option is intended for auth modules only).

use_mapped_pass - This argument is not currently supported by any of the modules in the Linux-PAM distribution because of possible consequences associated with U.S. encryption exporting restrictions.

expose_account - In general the leakage of some information about user accounts is not a secure policy for modules to adopt. Sometimes information such as user names or home directories, or preferred shell, can be used to attack a user’s account. In some circumstances, however, this sort of information is not deemed a threat: displaying a user’s full name when asking them for a password in a secured environment could also be called being ‘friendly’. The expose_account argument is a standard module argument to encourage a module to be less discrete about account information as it is deemed appropriate by the local administrator.

Any line in (one of) the configuration file(s), that is not formatted correctly, will generally tend (erring on the side of caution) to make the authentication process fail. A corresponding error is written to the system log files with a call to syslog(3).

Directory based configuration

More flexible than the single configuration file, it is possible to configure libpam via the contents of the /etc/pam.d/ directory. In this case the directory is filled with files each of which has a filename equal to a service-name (in lower-case): it is the personal configuration file for the named service.

The Secure Console Port Server SSH Linux-PAM was compiled to uses both /etc/pam.d/ and /etc/pam.conf in sequence. In this mode, entries in /etc/pam.d/ override those of /etc/pam.conf.

The syntax of each file in /etc/pam.d/ is similar to that of the /etc/pam.conf file and is made up of lines of the following form:

```
module-type   control-flag   module-path   arguments
```

The only difference being that the service-name is not present. The service-name is of course the name of the given configuration file. For example, /etc/pam.d/login contains the configuration for the login service.

Example configuration file entries

This section gives some examples of entries that can be present in the Linux-PAM configuration file. As a first attempt at configuring your system you could do worse than to implement these.

Default policy

If a system is to be considered secure, it had better have a reasonably secure ‘OTHER’ entry. The following is a paranoid setting (which is not a bad place to start!):

```
#
# default; deny access
#
OTHER   auth     required     pam_deny.so
OTHER   account  required     pam_deny.so
OTHER   password required     pam_deny.so
OTHER   session  required     pam_deny.so
```

Whilst fundamentally a secure default, this is not very sympathetic to a misconfigured system. For example, such a system is vulnerable to locking everyone out should the rest of the file become badly written.

The module pam_deny not very sophisticated. For example, it logs no information when it is invoked so unless the users of a system contact the administrator when failing to execute a service application, the administrator may go for a long while in ignorance of the fact that his system is misconfigured.

The addition of the following line before those in the above example would provide a suitable warning to the administrator.

```
#
# default; wake up! This application is not configured
#
OTHER   auth     required       pam_warn.so
OTHER   password required       pam_warn.so
```

Having two “OTHER auth” lines is an example of stacking.

On a system that uses the /etc/pam.d/ configuration, the corresponding default setup would be achieved with the following file:

```
#
# default configuration: /etc/pam.d/other
#
auth     required       pam_warn.so
auth     required       pam_deny.so
account  required       pam_deny.so
password required       pam_warn.so
password required       pam_deny.so
session  required       pam_deny.so
```

On a less sensitive computer, one on which the system administrator wishes to remain ignorant of much of the power of Linux-PAM, the following selection of lines (in /etc/pam.conf) is likely to mimic the historically familiar Linux setup.

```
#
# default; standard UNIX access
#
OTHER   auth     required       pam_unix_auth.so
OTHER   account  required       pam_unix_acct.so
OTHER   password required       pam_unix_passwd.so
OTHER   session  required       pam_unix_session.so
```

In general this will provide a starting place for most applications.

Secure Console Port Server SSH Default pam.conf file

In addition to the normal applications login, su, sshd, passwd, and pppd Black Box also has made portslave a PAM-aware application. The portslave requires four services configured in the pam.conf. They are local, remote, radius, and tacplus. The portslave PAM interface takes any parameter needed to perform the authentication in the serial ports from the file pslave.conf. The pslave.conf parameter all.authtype determines which service(s) should be used.

```
# ————————————————————————————————————#
# /etc/pam.conf
#
#
#
# Last modified by Andrew G. Morgan <morgan@kernel.org>
#
# ————————————————————————————————————#
# $Id: pam.conf,v 1.2 2001/04/08 06:02:33 agmorgan Exp $
# ————————————————————————————————————#
# serv.   module    ctrl         module [path]  ...[args..]
#
# name type      flag
#
# ————————————————————————————————————--—#
```

```
#
# The PAM configuration file for the ‘tacplus’ service
#
tacplus   auth       requisite    pam_securetty.so
tacplus auth       required     pam_tacplus.so encrypt
tacplus account    required     pam_tacplus.so encrypt service=ppp protocol=lcp
tacplus session    required     pam_tacplus.so encrypt service=ppp protocol=lcp

#
# The PAM configuration file for the ‘radius’ service
#
radius auth       requisite  pam_securetty.so
radius  auth       required   pam_radius_auth.so
radius  account    required   pam_radius_auth.so
radius  session    required   pam_radius_auth.so

#
# The PAM configuration file for the ‘local’ service
#
local  auth       requisite  pam_securetty.so
local   auth       required   pam_unix.so
local   account    required   pam_unix.so
local   password   required   pam_unix.so md5 use_authtok
local   session    required   pam_unix.so

#
# The PAM configuration file for the ‘remote’ service
#
remote auth       required  pam_permit.so
remote account    required  pam_permit.so
remote password   required  pam_permit.so
remote session    required  pam_permit.so

#
# The PAM configuration file for the ‘login’ service
#
login  auth       requisite  pam_securetty.so
login  auth       required   pam_unix.so
login  auth       optional   pam_group.so
login  account    requisite  pam_time.so
login  account    required   pam_unix.so
login  password   required   pam_unix.so md5 use_authtok
login  session    required   pam_unix.so
login   session    required   pam_limits.so

#
# The PAM configuration file for the ‘xsh’ service
#
sshd   auth       required   pam_unix.so
sshd   auth       optional   pam_group.so
sshd   account    requisite  pam_time.so
sshd   account    required   pam_unix.so
sshd   password   required   pam_unix.so md5 use_authtok
sshd   session    required   pam_unix.so
sshd    session    required   pam_limits.so

#
# The PAM configuration file for the ‘passwd’ service
#
passwd password   required   pam_unix.so md5
#
```

```
# The PAM configuration file for the ‘samba’ service
#
samba  auth       required   pam_unix.so
samba  account    required   pam_unix.so
#
# The PAM configuration file for the ‘su’ service
#
su  auth       required   pam_wheel.so
su  auth       sufficient pam_rootok.so
su  auth       required   pam_unix.so
su  account    required   pam_unix.so
su  session    required   pam_unix.so

#
# Information for the PPPD process with the ‘login’ option.
#
ppp     auth    required    pam_nologin.so
ppp     auth    required    pam_unix.so
ppp     account required    pam_unix.so
ppp     session required    pam_unix.so

#
# The PAM configuration file for the ‘other’ service
#
other  auth       required   pam_warn.so
other  auth       required   pam_deny.so
other  account    required   pam_deny.so
other  password   required   pam_warn.so
other  password   required   pam_deny.so
other  session    required   pam_deny.so
```

Reference

The Linux-PAM System Administrators' Guide

APPENDIX K TIMEZONE

The content of the file /etc/TIMEZONE can be one of two formats. The first format is used when there is no daylight saving time in the local time zone:

```
std offset
```

The std string specifies the name of the time zone and must be three or more alphabetic characters. The offset string immediately follows std and specifies the time value to be added to the local time to get Coordinated Universal Time (UTC). The offset is positive if the local time zone is west of the Prime Meridian and negative if it is east. The hour must be between 0 and 24, and the minutes and seconds 0 and 59.

The second format is used when there is daylight saving time:

```
std offset dst [offset],start[/time],end[/time]
```

There are no spaces in the specification. The initial std and offset specify the standard time zone, as described above. The dst string and offset specify the name and offset for the corresponding daylight savings time zone. If the offset is omitted, it defaults to one hour ahead of standard time.

The start field specifies when daylight savings time goes into effect and the end field specifies when the change is made back to standard time. These fields may have the following formats:

- Jn This specifies the Julian day with n between 1 and 365 February 29 is never counted even in leap years.
- n This specifies the Julian day with n between 1 and 365. February 29 is counted in leap years.
- Mm.w.d This specifies day d (0 <= d <= 6) of week w (1 <= w <= 5) of month

m (1 <= m <= 12). Week 1 is the first week in which day d occurs and week 5 is the last week in which day d occurs. Day 0 is a Sunday.

The time fields specify when, in the local time currently in effect, the change to the other time occurs. If omitted, the default is 02:00:00.

In the example below:

GST+7DST+6M4.1.0/14:30.M10.5.6/10

The daylight saving time starts on the first Sunday of April at 2:30 pm and it ends on the last Saturday of October at 10:00 am.